Universidad Carlos III de Madrid

Escuela Politécnica Superior

Aplicación Web para la gestión de empresas de voluntariado

Proyecto Fin de Carrera

Ingeniería Técnica Informática

Autor: David Hernández Ruiz

Tutor: Fernando Paniagua Martín

Julio 2009

Agradecimientos

A mi familia y a todos los que han colaborado en la realización del proyecto.

A Fernando Paniagua por su paciencia infinita.

Resumen

La gestión de la información corporativa por parte de una empresa es una ardua y compleja tarea que normalmente se facilita a través de aplicaciones de escritorio, orientadas hacia usuarios con poca experiencia y conocimiento sobre el manejo de bases de datos. Sin embargo, aún existen empresas que no disponen de este tipo de aplicaciones o, disponiendo de ellas, desean gestionar dicha información a través de tecnologías distintas, como puede ser Internet.

Actualmente, se dispone de las tecnologías necesarias para poder llevar a cabo este tipo de desarrollos orientados a su publicación en Internet, las cuales permiten crear sistemas capaces de interactuar con el usuario, gestionando los datos de una forma sencilla y amigable. Se permite así a un usuario sin el conocimiento necesario un manejo sencillo de la información, requiriendo un desarrollo previo por parte de personal cualificado.

En este trabajo, se ha realizado el análisis, diseño e implementación de una aplicación Web que facilite la gestión de la información de una empresa intermediaria que basa su actividad en el mercado de los seguros, dentro del ámbito del voluntariado. La solución tecnológica elegida se ha desarrollado en Java, estando el soporte de datos delegado en un sistema gestor de bases de datos MySQL, siendo uno de los aspectos más relevantes la realización de una sencilla interfaz de gestión.

Como añadido, se ha incluido la generación de correos y documentos PDF para agilizar el intercambio de información.

El sistema se ha diseñado pensando en la reusabilidad, con el fin de poder ser utilizado en posteriores aplicaciones con conceptos similares, sin dejar de cumplir todas las necesidades y requisitos que el proyecto requería.

Abstract

The management of corporate information from a company is an arduous and complex task that would normally be provided through desktop applications, oriented towards users with little experience and knowledge of database management. However, there are still companies that do not have this kind of applications, or you have them, want to manage this information across different technologies, such as the Internet.

Currently, you have the technologies necessary to carry out this type of development-oriented publication on the Internet, which allow you to create systems capable of interacting with the user, managing the data in a simple and friendly. It allows a user without the need for a simple information, requiring a previously developed by qualified personnel.

In this work, has done the analysis, design and implementation of a Web application that facilitates the information management basa an intermediary company that operates in the insurance market within the field of volunteerism. The technology solution chosen was developed in Java, while the data support a delegated management system MySQL, one of the most important aspects of conducting a simple management interface.

In addition, has included the generation of PDF documents and emails to accelerate the exchange of information.

The system is designed on reusability, to be used in subsequent applications with similar concepts, while fulfilling all the needs and requirements that the project required.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

En este primer capítulo se explican las motivaciones que han llevado a la realización de este trabajo así como los objetivos fijados en el mismo. Por otra parte se detalla el contenido de la memoria, explicando los distintos apartados que la componen.

1.1 Motivación del proyecto

La información siempre ha sido uno de los principales activos de las empresas y su gestión uno de los procesos más importantes. Dichas empresas han utilizado desde libros de registro tradicionales hasta sistemas gestores de bases de datos como Oracle o MS-Access como sistemas de almacenamiento de datos, cada uno con sus características, pero todos con el mismo fin: llevar un control y clasificación sobre aquella información de se desea guardar.

Uno de los problemas de los sistemas antiguos es su capacidad y modo de recuperación de la información. Pongamos un ejemplo de un organismo estatal, como puede ser el registro civil. Antiguamente la información de las partidas de nacimiento se almacenaba en libros de registro, clasificados alfabética y cronológicamente en estanterías cuya capacidad tenía un límite. Suponiendo el caso en el que alguien quiera acceder a esa información, deberá dirigirse a la estantería correcta, y posteriormente al archivo concreto que guarde el libro de registro que contiene la partida deseada, lo que conlleva un tiempo innecesario en la era actual. En la actualidad este problema ya ha sido subsanado o está en proceso de serlo, mediante la utilización de bases de datos informatizadas. Éstas permiten almacenar tanta información como sea necesaria, lo que conlleva un ahorro importante de espacio físico, así como proporciona facilidad y agilidad al proceso de consulta de las partidas de nacimiento. Todo ello supone una gran inversión inicial, pero una vez implantado, el ahorro de los costes es incalculable.

En la actualidad, existen gran cantidad de casos como este, empresas que por unos u otros motivos no han actualizado su sistema de gestión de base de datos y siguen con sus antiguas técnicas para almacenar la información.

Una de las motivaciones del proyecto, fue la petición por parte de una empresa de seguros de la realización de una aplicación. La función de esta empresa de seguros es la comercialización de productos de seguros diseñados especialmente para el mundo de la discapacidad y el tercer sector.

Hasta ahora, esta empresa se encargaba de poner en contacto y administrar los voluntarios de las asociaciones ligadas a ella, con una sociedad de seguros, teniendo para ello que gestionar todos los contratos, certificados y demás documentos manualmente o a través de ficheros anticuados. La solución posible a este problema que la empresa presentaba, se planteó hacia una mejora en los tiempos de generación de documentos y altas de los voluntarios. Se quería informatizar estos procesos, y poder delegar ciertas tareas en las mismas asociaciones a las que pertenecían los voluntarios. A su vez, querían tener una base de datos donde almacenar toda la información sobre las asociaciones y voluntarios, y que, a través de una sencilla interfaz, pudiesen gestionarla. La opción era perfecta para realizar el desarrollo de una aplicación Web: se trataba de una solución muy buena para el problema que planteaban y daba la oportunidad de desarrollar por completo, de principio a fin, un proyecto realmente útil y remunerado.

El principal requisito por parte de la empresa era poder agilizar la emisión de certificados de los voluntarios dados de alta, ya que para ellos suponía un gran gasto de tiempo, puesto que por cada voluntario dado de alta debían rellenar un documento Word y mandarlo a la dirección de contacto de la asociación a la que pertenecía el voluntario.

Su base de datos estaba informatizada, pero consistía en una carpeta dentro de uno de los ordenadores de la empresa, la cual contenía un archivo de asociaciones y varios con los voluntarios de cada una, con un descontrol total en el modelo de datos.

Otra de las peticiones era disponer de módulos informatizados donde las asociaciones pudiesen gestionar sus propios voluntarios, descargando así la cantidad de trabajo realizada por esta empresa, ya que lo antes hacían ellos en un largo periodo de tiempo, ahora lo harían las asociaciones en unos minutos.

El análisis, diseño y desarrollo de un sistema Web que cumpla con las necesidades explicadas en este capítulo ha constituido la motivación de este trabajo fin de carrera.

1.2 Objetivos

Dadas las motivaciones expuestas en el anterior apartado, se desea realizar una aplicación Web para, utilizando Internet como medio de difusión, cubra por completo las necesidades de la empresa demandante.

El primer objetivo será realizar una aplicación segura y reutilizable a través del patrón Modelo-Vista-Controlador (MVC), el cual permite implementar de manera independiente las diferentes capas que constituyen dicha aplicación. Este modelo proporciona una organización del diseño del software que aporta independencia entre las capas del mismo (presentación, control y modelo), lo cual permite aislar los problemas propios de cada capa y resolverlos sin tener en consideración los demás. Por otra parte la independencia entre capas hace que sean fácilmente sustituibles, evitando dependencias y facilitando la reusabilidad.

Como ya se ha indicado, el diseño de la aplicación se basa en la utilización de dicho patrón MVC, que permite separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Este patrón, está muy extendido en aplicaciones Web, donde la vista es una página en HTML, JSP, o cualquier otro lenguaje de programación Web. El modelo es el sistema de gestión de la base de datos y la lógica de control, mientras que el controlador se hará cargo decidir qué componente o componentes resuelve cada uno de los problemas, ya sea de lógica de negocio o de presentación.

Uno de los mayores problemas de los sistemas de gestión de bases de datos es la dificultad que tienen en su manejo, muchos de ellos resultan muy complejos para un usuario inexperto. Muestran una interfaz poco intuitiva y difusa, lo que dificulta aún más su aprendizaje. Hay que suponer que la mayoría de los usuarios no tendrán conocimiento alguno sobre estos sistemas, y que lo que buscarán en ellos serán sencillez y claridad en su uso. Así pues, el segundo objetivo, es acercar al usuario a través de una sencilla interfaz, un sistema de gestión de base de datos, eliminando todo aquel proceso que pueda ser complicado y costoso para el usuario. La interfaz se programará en JSP, un lenguaje que permite generar contenido dinámico en las páginas Web. Será el encargado tanto de recoger los datos introducidos por el usuario, como de mostrarlos por pantalla una vez procesados por el sistema. Se realizará lo más intuitivo posible, buscando que el tiempo de aprendizaje sea muy bajo, ofreciendo al

usuario un manual de ayuda y gestionando todos aquellos errores que cometa en el uso de la aplicación.

La funcionalidad básica de la aplicación será la siguiente:

- Gestionar las altas, bajas y modificaciones de las asociaciones
- Gestionar las altas, bajas y modificaciones de los voluntarios.
- Generar los certificados, contratos y correos necesarios para la gestión de los seguros.
- Consultar cualquier información referente a las asociaciones y voluntarios.

Finalmente, el último objetivo y más importante para el cliente, será reducir lo máximo posible el tiempo de gestión de un voluntario o asociación. El cliente busca que la aplicación genere automáticamente todo aquello que anteriormente tramitaban de una forma manual y costosa. La generación de certificados se realizaba de uno en uno, teniendo que crear un nuevo archivo por cada voluntario, rellenarlo y enviarlo a la correspondiente dirección de correo. Igualmente pasaba con los contratos, libros de registro, etc.…La aplicación resolverá de la manera más eficiente posible estos problemas, liberando al usuario de esa carga extra de trabajo que conllevaba estos procesos.

1.3 Contenido de la memoria

En esta memoria se intenta reflejar el contenido del desarrollo de una aplicación que trata de dar una solución a un problema a un sistema de gestión de bases de datos. Se enseñan tanto los contenidos teóricos relaciones con el motivo de la aplicación, como aquellos procesos concretos desarrollador para hacer efectivos los objetivos fijados inicialmente.

La memoria está organizada en los siguientes apartados principales:

Resumen: Introducción a la situación existente en la gestión de las bases de datos, así como una breve descripción del trabajo realizado.

Motivación del proyecto: Se explican las razones que han llevado a realizar este proyecto, por qué puede resultar útil, y los posible problemas que puede solucionar.

Objetivos: Declaración de intenciones en la que se indican, de forma concreta, qué productos o conclusiones se desea obtener con la realización de este proyecto.

Estado del Arte: En este apartado se presenta información relacionada con el trabajo. Se explica qué significan los conceptos y tecnologías concretas utilizadas. También se muestra información relacionada con soluciones similares o alternativas.

Metodología: Basándose en los estándares mencionados posteriormente, se realiza este apartado. Forma parte de él también el ciclo de vida escogido para el desarrollo.

Análisis: Contiene la descripción completa de la aplicación, junto con los requisitos detallados que demanda el cliente.

Diseño: Este apartado contiene todos los diseños que se han realizado para la aplicación. Diseño de clases, diagramas de uso, etc.

Desarrollo del Proyecto: Compuesto por un conjunto de apartados en los que se trata de explicar los procesos realizados, cómo funcionan y qué objetivos tienen. En estos apartados se detallan técnicamente las soluciones encontradas así como los problemas surgidos durante el desarrollo y la forma en que estos se han solucionado.

Pruebas: Se ha realizado una batería de pruebas formales y son presentadas en este apartado con el fin de certificar el correcto funcionamiento de los procesos desarrollados.

Conclusiones y Trabajos Futuros: Indicaciones de cómo y en qué puede el proyecto evolucionar en etapas futuras.

Manual de Instalación: Guía de instalación de la aplicación.

Manual de Usuario: En el manual de usuario se detalla la funcionalidad implementada en la aplicación así como el modo en que se ha de utilizar.

Bibliografía: Relación de fuentes documentales ya sean libros, otros trabajos o recursos obtenidos de Internet.

2 Estado del arte

En este capítulo se repasan los conceptos utilizados en la realización de la aplicación. En primer lugar se expone una visión global del comercio electrónico así como la evolución de estos últimos años en el desarrollo de aplicaciones Web. Posteriormente se explican brevemente las tecnologías utilizadas como Java, MySQL o JSP, así como otras secundarías que han resultado ser claves en el desarrollo.

Se representan también otras soluciones alternativas para la realización de la aplicación, exponiendo sus ventajas e inconvenientes si se hubiese adoptado cualquiera de ellas.

2.1 Evolución de las aplicaciones Web

Se denomina una aplicación Web a aquellas aplicaciones que permiten al usuario interactuar con ella a través de un servidor Web, Internet o a través de una intranet mediante una navegador. Su facilidad en el manejo, actualización y mantenimiento, las han convertido en la actualidad en unas de las aplicaciones mas populares y extendidas.

Cabe destacar que una de las funciones más importantes de estas aplicaciones es el intercambio de información de una manera activa con el usuario. El usuario interactúa con la aplicación, pudiendo acceder a todo tipo de información, como pueden ser blogs, tiendas online, webmail, etc.

2.1.1 Pasado y presente

En un principio, la Web era tan solo una colección de sencillas páginas estáticas, donde poder obtener cierta información para su consulta o descarga. Más tarde, el primer paso hacia la interactividad con el usuario fueron los "Common Gateway Interface" (CGI), que definía un mecanismo mediante el que se podía poner en contacto al usuario con el servidor, intercambiando información entre ellos.

A día de hoy se sigue utilizando los CGI, puesto que supone una manera sencilla de programación y dan total libertad para utilizarse con otro lenguaje de programación.

Sin embargo, el funcionamiento de los CGI tienen un grave problema, cada vez que se recibe una petición, el servidor debe lanzar un nuevo proceso para ejecutar el programa. Como la mayoría de los CGI estaban escritos en lenguajes de programación interpretados, Python, Perl, C, el servidor estaba sometido a una gran carga.

Debido a esta situación, se empiezan a desarrollar otras alternativas para solucionar este problema principal. Surgen dos vías:

- Se diseñan nuevos sistemas de ejecución de módulos mejor integrados con el servidor, evitando así, la instanciación y ejecución de varios programas.

- Se dota a los servidores de un intérprete de algún lenguaje de programación para agilizar el proceso de ejecución y reducir el intervalo de respuesta.

Se experimenta así un incremento del número de arquitecturas y lenguajes que optan por alguna de estas dos vías que permitan desarrollar aplicaciones Web.

Las más útiles y utilizadas permiten mezclar dos sistemas, un lenguaje integrado que permita al servidor interpretas comandos incrustados en las páginas, y un sistema de ejecución de programas mejor enlazado con el servidor para evitar los antiguos problemas de los CGI.

En la actualidad, una de las más potentes herramientas utilizadas para el desarrollo de aplicaciones Web es la seguida por Sun Microsystems, Java, que consta de dos componentes:

- Un lenguaje que permite la incrustación de código en las páginas HTML que el servidor convierte en programas ejecutables, **JSP**.

- Un método que realiza una función similar a los CGI y muy ligado al servidor, obteniendo así un rendimiento muy superior, **Java Servlet**.

Otro de los sistemas muy utilizado es el lenguaje PHP, que igualmente, permite incrustar código en las páginas HTML con sintaxis derivada de C y Perl. Es una herramienta sencilla y potente, que en determinados desarrollos puede ser muy útil.

2.2 Comercio electrónico

2.2.1 Introducción

Los métodos de comercio han ido evolucionando paralelamente a lo largo de la historia a la evolución humana.

A principios de los años veinte, apareció en Estados Unidos la venta por catálogo, impulsado principalmente por empresas mayoristas. Consiste básicamente en la venta de productos a través de un catálogo donde se muestran las fotos de estos, suponiendo un sistema revolucionario para su época. Mejoraba la compra de los clientes, los que podían elegir los productos a comprar desde su casa, sin presión alguna por parte del vendedor. Además, suponía una ampliación Además. Facilitaba la distribución en zonas rurales, donde los clientes debían hacer un gran desplazamiento para visitar la tienda. La tarjeta de crédito supuso un gran impulso para este tipo de venta, permitiendo el anonimato entre cliente y servidor.

A partir de los años setenta, aparecen las primeras relaciones comerciales que utilizaban el ordenador para el intercambio de datos. No tenía ningún estándar por lo que trataron de fijar alguno para realizar este intercambio, lo que trajo consigo una mejora en los procesos de fabricación en el ámbito privado.

A mediados de 1980, gracias en gran parte a la televisión, surgió una nueva forma de distribución de los catálogos, llamada venta directa. Se crea así un nuevo método que permite mostrar los productos con un mayor realismo y dinamismo, pudiendo resaltas las características con mayor facilidad. La compra de estos productos se hacían mediante una linea telefónica y los pagos a través de una tarjeta de crédito.

En 1995, los países integrantes del G7/G8, crean la iniciativa "Un mercado global para PYMES", con el propósito de acelerar el uso del comercio electrónico entre las empresas.

Ventajas para los clientes.

- Mejorar en la distribución. El cliente puede ponerse en contacto directo con el proveedor del producto, evitando así a los intermediarios, por lo que los costos tienden a cero. Esta situación puede llegar a reducir los canales de

comercialización, permitiendo que su distribución sea eficiente reduciendo sobrecostos. También se reducen tiempos en las transacciones comerciales.

- Comunicaciones de mercadeo. Actualmente, la mayoría de las empresas disponen de una página Web donde informar de los productos o servicios a sus clientes. Esta situación potencia las relaciones con los clientes, traduciéndose en una facilidad de mercadeo y de soporte al cliente nunca imaginado. Los clientes pueden utilizar la Web las 24 horas del día, y estar contactados con el vendedor a través de correo electrónico. Los sitios Web más sencillos utilizan correo electrónico para establecer una comunicación cliente-servidor. En otros sitios más sofisticados, se utilizan formularios con el objeto de que desarrollen una relación continua con la compañía.

- Beneficios operacionales. El uso de la Web, reduce tiempo, errores y sobrecostos. Los proveedores disminuyen sus costos al acceder directamente a la base de datos. Se facilitará la creación de mercados y nuevos segmentos, especialmente en los geográficamente remotos.

Clasificación del comercio electrónico

El comercio electrónico puede dividirse en cuatro categorías según sus entidades relacionadas.

- **Compañía – Compañía (B2B).** Una compañía hace uso de una red para ponerse en contacto con sus proveedores y gestionar sus facturas y pagos correspondientes. Este tipo de comercio ha sido utilizado muchos años a través de EDI sobre redes privadas o de valor agregado.

- **Compañía – Cliente (B2C).** El cliente realiza sus compras a través de los sitios Web de las compañías. Este tipo ha tenido gran aceptación debido a la existencia de grandes centros comerciales por todo Internet.

- **Compañía – Administración (B2G).** Se refiere a todas las transacciones llevadas a cabo entre las compañías y las organizaciones del gobierno. Todavía esta en sus inicios, pero a medida que el gobierno empiece a hacer uso de este comercio, el comercio electrónico alcanzara su mayor potencial

- **Cliente – Administración (C2G).** Aun no ha nacido, sin embargo, el gobierno tiene previsto esta expansión para efectuar interacciones electrónicas como serían pagos de asistencia social o devolución de impuestos.

Riesgos

Se pueden presentar los siguientes deficientes tanto por su naturaleza como de su tecnología.

- **Entorno empresarial y tecnológico cambiante.** Empresas y clientes quieten tener flexibilidad para cambiar de socios sociales, plataformas y redes. Una empresa como mínimo deberá disponer de un ordenador con conexión a Internet. Si desea involucrarse más, deberá introducir un sistema como EDI con sus proveedores y/o una intranet con sus diversas sedes.

- **Privacidad y seguridad.** La mayoría de los usuarios desconfía de las transacciones Web como canal de pago. Existen sistemas que funcionan correctamente para operaciones comerciales muy altas, sin embargo, los problemas se centran en las operaciones pequeñas.

- **Cuestiones legales, políticas y sociales.** Validez de la firma, no repudio, legalidad de un contrato electrónico, etc.

Tecnologías empleadas

El comercio electrónico utiliza un amplio rango de tecnologías.

- Intercambio Electrónico de Datos (EDI-Electronic Data Interchange)
- Correo Electrónico (E-mail o Electronic Mail)
- Transferencia Electrónica de Fondos (EFT- Electronic Funds Transfer)
- Aplicaciones Internet: Web, News, Gopher, Archie
- Aplicaciones de Voz: Buzones, Servidores
- Transferencia de Archivos

- Diseño y Fabricación por Computadora (CAD/CAM)
- Multimedia
- Tableros Electrónicos de Publicidad
- Videoconferencia

2.2.2 Evolución y características del modelo B2B

Al principio de los años setenta, en Estados Unidos, las innovaciones como la Transferencia Electrónica de Fondos (EFT) y el Intercambio Electrónico de Datos (EDI), cambiaban la forma de operar de las grandes compañías, instituciones y algunas pequeñas empresas.

En los años noventa, y con los primeros navegadores de Internet, permitían visionar páginas en formato HTML. El objetivo principal era publicar textos planos que permitían compartir información con usuarios en distintas ubicaciones geográficas.

En 1996, compañías estadounidenses dedicadas a los servicios de mensajería, comenzaron a llevar sus bases de datos a Internet, para beneficio de sus clientes, rompiendo con el paradigma de que dichas bases no debían salir de la compañía por motivos de seguridad. Los clientes comenzaron así a realizar y esperar envíos, teniendo la posibilidad de conocer en tiempo real la ubicación de sus paquetes.

Fue asi como da comienzo una nueva forma de hacer negocios entre compañías, los modelos B2B del comercio electrónico, caracterizándose porque su aplicación se realiza entre empresas.

En el B2B, la empresa que compra puede utilizar los bienes o servicios para ofrecerlos al consumidor final, o bien para producir otro bien y ofrecerlo a otro mercado.

Algunas de las aplicaciones de los sistemas B2B son las siguientes:

- Vender y distribuir productos a otros negocios.
- Llevar a cabo de forma sistemática el suministro de bienes.

- Proveer la logística necesaria para colocar los productos correctos en un tiempo óptimo dentro de una organización.
- Ofrecer un soporte de ejecución de actividades de mercadotecnia y publicidad.
- Ofrecer el adecuado servicio para diferentes áreas funcionales.
- Generar información para pronósticos.

Ventajas del modelo B2B.

- Permite automatizar las relaciones comerciales entre empresas (Proveedores y Compradores).
- Rapidez y seguridad en las comunicaciones.
- Integración directa de los datos de la transacción en los sistemas informáticos de la empresa.
- Posibilidad de incrementar el número de de ofertas y demandas.
- No existe trato con el comprador o vendedor, por lo que no existen tratos de favor.
- Abaratamiento del proceso, agilizando el proceso de negociación y reduciendo las visitas comerciales.

Ventajas para el comprador

- Aumento de sus fuentes de suministro.
- Disminuye el plazo entre solicitud y recepción.
- Reduce los costos por transacción.
- Descienden las compras fuera de contrato.
- Aumento de control sobre el proceso de aprovisionamiento.

Ventajas para el proveedor

- Incremento de la base de clientes potenciales.

- Aumento de la fidelidad de sus clientes gracias a la reducción de barreras de salida.
- Reducción de la carga administrativa.

Seguridad en las transacciones B2B

El importe unitario en las transacciones B2B es muy alto, lo que hace a este tipo de comercio online especialmente apetecible para piratas informáticos. Es aquí donde se requieren productos de autenticación fuertes que permitan dotar de la seguridad y flexibilidad a los mecanismos transaccionales de Internet.

Para ello, los "Códigos de Conducta" para el comercio electrónico B2B, deberían contener al menos:

- El ámbito del código de conducta. El código habría de indicar quien ha participado en la elaboración del mismo, además de indicar, bajo qué formulas comerciales es aplicable.
- Información sobre el e-market. Para que las empresas puedan distinguir entre mercado acreditados y económicamente solventes, se ha de facilitar a las mismas, información real acerca de los aspectos financieros y legales de cada mercado electrónico. Es fundamental el papel que juega la transparencia en este sentido.
- Información sobre los participantes. Podría vulnerar ciertas reglas sobre la competencia y las leyes del libre mercado, pero la información acerca de las empresas participantes es fundamental para acreditar cierto nivel de seguridad.
- Información sobre el modelo procedimental en las transacciones. Debería definir, cuando y como cada contrato es perfeccionado y que criterios son relevantes para concluir una transacción.
- Información sobre el mecanismo de fijación de precios. La claridad en este apartado ayudara a los participantes a eliminar potenciales preocupaciones acerca de practicas que puedan manipular los precios siempre y cuando las leyes referentes a la libertad del mercado sean respetadas.

- Información sobre protección de la privacidad y confidencialidad. Se deberían incluir mecanismos que aseguren a los usuarios que la información personal proporcionada esta perfectamente asegurada de acuerdo con la Ley de Protección de Datos.

- Información sobre fórmulas de resolución de conflictos. Es importante que se pueda acudir a un mecanismo alternativo de resolución de conflictos en caso de un eventual conflicto, siendo también de gran importancia que se especifiquen los procedimientos a seguir.

- Ley aplicable. Es importante que las partes hayan determinado con anterioridad bajo qué ley han de solucionar los posibles conflictos que surjan durante la transacción.

2.2.2.1 Intercambio electrónico de documentos (EDI)

Con el fin de acreditar la validez legal y fiscal de los documentos intercambiados surge la opción de utilizar EDI.

EDI es el intercambio electrónico de datos entre sistemas de información, por medios electrónicos, de datos estructurados de acuerdo con normas de mensajes acordadas. A través de EDI, las partes cooperan sobre un entendimiento claro y predefinido acerca de un negocio común.

Las interacciones tienen lugar por medio de aplicaciones informáticas que actúan como una interfaz con los datos locales, pudiendo intercambiar información comercial estructurada. EDI, establece como se estructuran los datos de los documentos electrónicos y define el significado comercial de cada elemento de datos. Para transmitir la información necesita un servicio de transporte adicional.

EDI respeta en todo momento la autonomía de las partes involucradas.

Los típicos campos de aplicación del EDI son el intercambio de información industrial, financiera, médica, administrativa o cualquier otro tipo similar de información estructurada. Esta información, se estructura en unos formatos que pueden ser procesados por aplicaciones informáticas.

La automatización de las interacciones por medio de EDI minimiza las transacciones sobre el papel y la intervención humana, pudiendo incrementar la eficiencia de las tareas diarias y mejorar las relaciones con agentes externos.

Los clientes potenciales de EDI son muchos, debido a que esta dirigido a empresas que se relacionan comercialmente.

- Sector Distribución (Supermercados, Proveedores).
- Sector de las Automotrices (Terminales, Proveedores y Concesionarios).
- Sector Farmacéutico (Farmacias, Laboratorios).
- Sector de la Administración Publica.
- Sector del Transporte y Turismo.

Transmisión de documentos entre empresas

El sistema tradicional en el que las empresas basan su transmisión de documentos, el papel, presenta dos grandes inconvenientes. En primer lugar, la lentitud. Documentos generados en potentes ordenadores que procesan información a gran velocidad, sufren retrasos por tener que procesarse de forma manual en las empresas de correos. Por este motivo, muchas empresas han sustituido el correo como medio de transmisión para enviar documentos, por el fax, lo que agiliza en buena medida la gestión. Sin embargo, existe otro problema, los diversos tipos de facturas, pedidos, hojas de precios, etc. Esta falta de normalización es causa de muchos errores administrativos.

A continuación se muestre el procedimiento convencional de transmisión de documentos administrativos entre empresas.

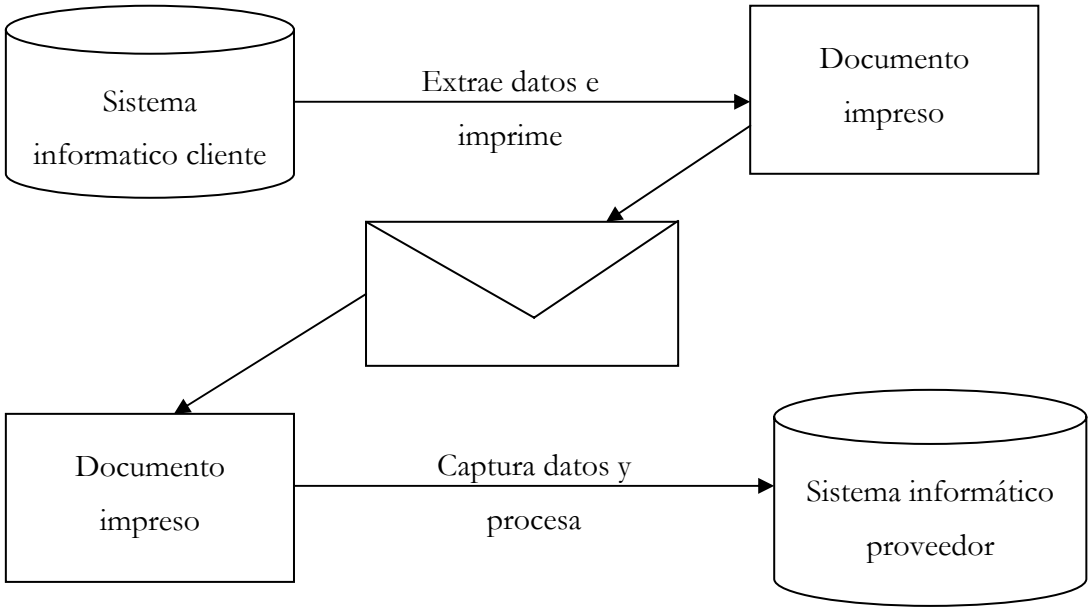

Ilustración 2-1 Transmisión de documentos convencional

La primera empresa consulta la base de datos y extrae e imprime la información necesaria. Esta información, se envía por correo electrónico a la otra empresa, quien debe de introducir de nuevo los datos en su sistema informático. En este sistema se producen redundancias, ya que los documentos que se imprimen en una empresa, son introducidos manualmente en el sistema informático de la otra.

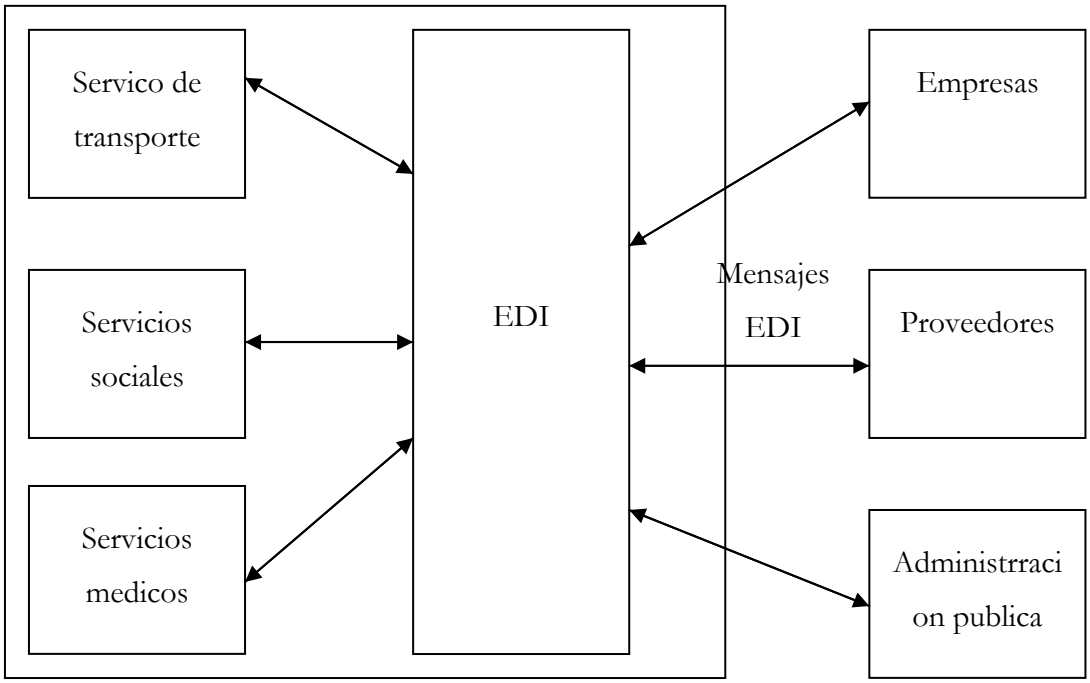

Ilustración 2-2 Transmisión de datos EDI

Esta figura representa a grandes rasgos lo que supone implantar un sistema EDI.

Consiste simplemente en implantar un procedimiento electrónico de transmisión de información. Las empresas se beneficiaran de este sistema con diversas formas de ahorro. En primer lugar un ahorro de tiempo, ya que la información viaja por redes de comunicación. Y en segundo lugar, se producen menos errores, ya que el proceso esta completamente automatizado.

Beneficios EDI

- Agilización de procesos comerciales.
- Disminución de errores en los documentos.
- Disminución de stocks debido a la facilidad de aplicación de técnicas "just in time".
- Ahorro de costos de administración.
- Mejora de la competitividad de la empresa que lo usa.

2.3 Conceptos técnicos

2.3.1 MVC

Se trata de un patrón de arquitectura de software que divide en tres componentes la creación de una aplicación. Por un lado los datos (modelo), por otro la interfaz de usuario (vista) y finalmente la lógica de control (controlador).

- **Modelo**: Representación específica con la que el sistema interactúa. Es el responsable de acceder a la capa de almacenamiento de datos, define las reglas de negocio y lleva un registro de las vistas y controladores del sistema.

- **Vista**: Define la interfaz de usuario, mostrándole la información del modelo. Recibe los datos del modelo y los muestra al usuario, tienen un registro de su controlador asociado y pueden ofrecer el servicio de actualización.

- **Controlador**: Responde a eventos y modifica la vista y el modelo. Recibe los eventos de entrada y contiene las reglas de gestión de estos.

El flujo de control suele seguir este procedimiento:

- El usuario interactúa de alguna manera con el interfaz.

- El controlador recibe la petición del usuario y la gestiona.

- El controlador accede al modelo y realiza la función requerida por el usuario (actualización, modificación….).

- La vista obtiene los datos y los muestra al usuario.

- La interfaz espera nuevas acciones por parte del usuario.

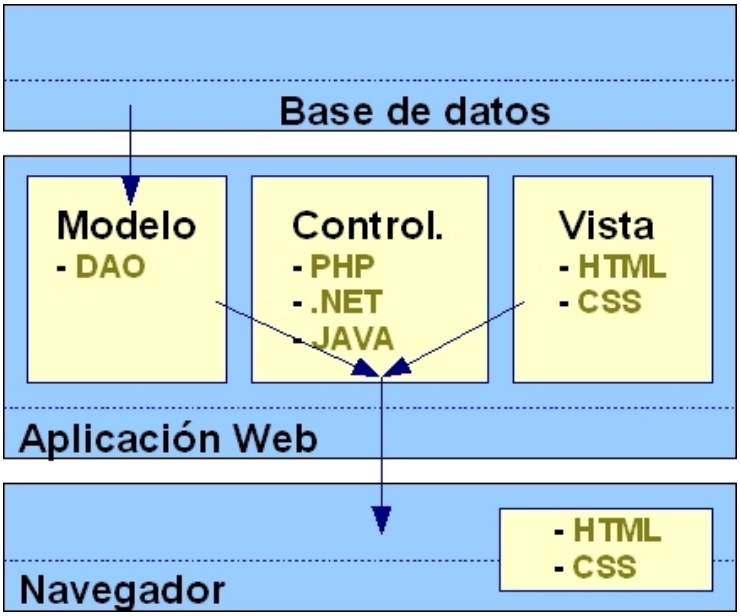

Ilustración 2-3 Modelo Vista Controlador

2.3.2 JAVA

Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. Permite una programación más sencilla que otros lenguajes orientados a objetos puesto que elimina herramientas de bajo nivel como puede ser la manipulación directa de punteros o memoria.

Generalmente es compilado en un bytecode, aunque es posible crear código máquina nativo. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque también existen dispositivos hardwares capaces de ejecutarlo directamente.

Sun Microsystems desarrollo en 1995 el compilador, la máquina virtual y las bibliotecas de clases, y desde entonces controla el desarrollo y la evolución del lenguaje a través del Java Community Process.

La mayoría de las tecnologías Java fueron liberadas entre el 2006 y 2007, convirtiéndose así prácticamente en su totalidad en software libre.

Java fue diseñado para ser:

- **Sencillo, orientado a objetos y familiar**. Fácil de aprender para los desarrolladores, orientado a objetos por ser una tecnología más madura y mejor para los sistemas distribuidos y cliente/servidor, y familiar por mantener gran parecido con C++, eliminando bastantes de sus complejidades.
- **Distribuido**. Java proporciona clases y herramientas para su uso en aplicaciones de red.
- **Robusto y seguro**. Robusto por simplificar la manipulación directa de punteros y gestión de memoria, y seguro para operar en los entornos de red.
- **Independiente de la arquitectura y portable**. El compilador de Java genera un bytecode, formato de código independiente de la plataforma, e interpretable por diversas plataformas de hardware y sistemas operativos. Portable por ser el mismo lenguaje en todas las plataformas, lo único que diferencia a estas es su Java Virtual Machine.
- **Alto rendimiento**. Java es un lenguaje interpretado, sin embargo, tiene en cuenta el rendimiento, y en las últimas versiones a puesto a disposición herramientas de optimización.
- **Interpretado, multi-hilo y dinámico**. Interpretado por generar un bytecode ejecutable en cualquier máquina con Java Virtual Machine. Java soporta múltiples hilos de ejecución. Proporciona mecanismos de carga dinámica y ejecución en la fase de enlazado.

Permite generar varios tipos de aplicaciones:

- Aplicaciones autónomas.
- Applets. Programas incrustados en otras aplicaciones.
- Servlets. Componentes de la parte de servidor de Java EE, genera respuestas a las peticiones de los clientes (detallado más adelante).
- Aplicaciones con ventanas. Usando la interfaz gráfica de Java.

El diseño de Java ofrece varios entornos de funcionamiento, entre los que se encuentran:

- Dispositivos móviles y sistemas empotrados.
- En navegador web.

- En sistemas de servidor.
- En aplicaciones de escritorio.

Para ejecutar Java, es necesario el JRE (Java Runtime Environment), que incluye la Java Vitual Machine y la Api. Esta incluida en cualquier versión de J2SE (Java 2 Estándar Edition), aunque si tan solo se quiere ejecutar la aplicación Java, con el JRE será suficiente, instala los plugins necesarios para su ejecución en los navegadores y sistemas operativos.

Para el desarrollo de aplicaciones, se necesita otro tipo de paquete, donde el J2SE es el más usado, aunque existen otro tipo de paquetes que permiten el desarrollo de otro tipo de aplicaciones, ejemplo de ello son:

- **Java SE**. Edición estándar de Java.
- **Java EE**. Aplicaciones distribuidas orientadas al entorno empresarial.
- **Java ME**. Dispositivos móviles, PDA, teléfono…
- **Java FX**. Permite crear aplicaciones dinámicas de nueva generación.

Cada una de ellas posee una API, conjunto de clases, interfaces y excepciones requeridas para el desarrollo de aplicaciones en cada plataforma.

A su vez, también existen paquetes creados externos a los oficiales, pero que amplían el uso y desarrollo del lenguaje (JavaMail, IText…).

En cuanto a las críticas posibles del lenguaje, se pueden encontrar algunas:

- No es un lenguaje estrictamente orientado a objetos.
- El código puede ser redundante en comparación con otros lenguajes.
- No dispone de operadores de sobrecarga definidos por el usuario.
- Rendimiento algo bajo comparado con otros lenguajes.

2.3.3 SERVLETS

Los servlets son módulos de Java que se ejecutan en el servidor o contenedor J2EE, expandiendo así las capacidades de los servidores Web. Se dice de ellos que son la siguiente

etapa de los CGI, ya que su función más típica es generar páginas web dinámicas a partir de los parámetros de la petición que envíe el navegador.

Los servlets implementan la interfaz javax.servlet.Servlet, permitiendo interpretar objetos del tipo HttpServletRequest/Response, contenedores de la información de la página que ha invocado al servlet.

HttpServletRequest representa el la comunicación desde el cliente al servidor, mientras que HttpServletResponse representa la comunicación inversa.

El ciclo de vida del servlet se puede dividir en las siguientes fases:

- El cliente realiza una petición.
- Si es la primera vez que se recibe la petición, el servlet ejecutará el método de inicio init(), inicializando las variables generales.
- Si no es la primera vez, cada petición se convierte en un nuevo hilo.
- Dependiendo de la petición, se ejecutará el método doGet() o doPost(), generando el resultado a devolver.
- Finalmente el servlet eliminará todos los datos generados llamando al método destroy().

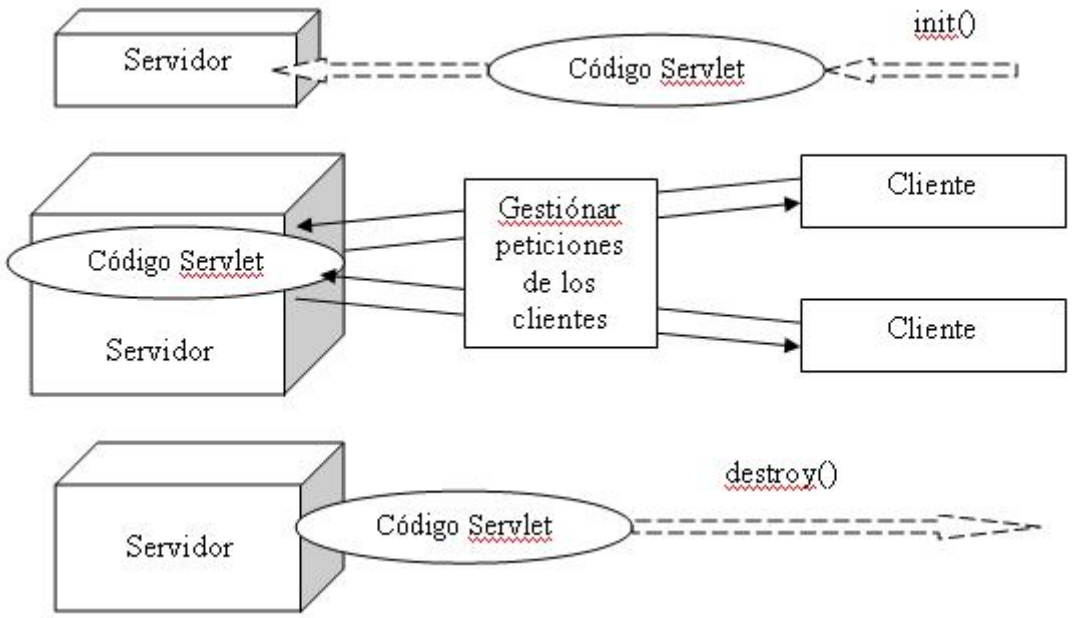

Ilustración 2-4 Diagrama Servlets

2.3.4 MySQL

MySQL es un sistema de gestión de base de datos relacional muy rápido, multihilo, multiusuario y robusto. Esta diseñado para entornos de producción críticos, con alta carga de trabajo.

Consta de una doble licencia, una totalmente gratuita bajo los términos de la licencia GNU Genral Public License u otra bajo una licencia comercial estándar de MySQL AB. Consulte.

Las principales características son:

- Es un sistema de gestión de bases de datos.
- Es un sistema de gestión de base de datos relacionales.
- Es Open Source.
- El servidor de base de datos es muy rápido, fiable y fácil de usar.
- Trabaja en entornos cliente/servidor o incrustados.
- Existe una gran cantidad de software que utiliza este sistema.

Al ser una base de datos muy rápida en la lectura, es ideal para utilizarla en la creación de aplicaciones web, muy utilizada en conjunto con PHP.

2.3.5 JSP

Es una tecnología de Java orientada a la creación de contenido dinámico para las páginas web. Se podrían considerar como una manera alternativa y sencilla de construir servlets, además de poder utilizarse como Vista, dentro del Modelo Vista Controlador, encargándose de recibir los parámetros desde el servlet y mostrándolos al usuario, y de recoger las peticiones del usuario y llevarlos al servlet.

Al tratarse de una tecnología Java, se puede utilizar en cualquier máquina siempre que tengan instalado la máquina Virtual de Java.

Permite agregar etiquetas XML, llamadas acciones, además de poder utilizar y crear bibliotecas de etiquetas JSP .

Un JSP es compilado a un programa en Java la primera vez que se invoca, de este programa se crea una clase que se ejecuta en el servidor como si se tratase de un servlet, siendo la diferencia con estos, que un JSP es una página Web con etiquetas y código Java incrustado (script), mientras que un servlet es un programa que recibe peticiones y genera respuestas a partir de ellas.

2.3.6 APACHE TOMCAT

Tomcat es un servidor web capaz de soportar Servlets y JSP. Funciona como un contenedor de servlets, pero no como un servidor de aplicaciones, pudiendo dividir los contenedores en:

- Contenedores de Servlets independientes. Son una parte integral del servidor web. Sin embargo, no es lo más común, ya que la mayoría de servidores no están basados en Java.
- Contenedores de Servlets dentro de proceso. Es una combinación de plugin para el servidor y una implementación de contenedor Java. El plugin abre una maquina virtual de Java dentro del espacio de direcciones del servidor permitiendo que el contenedor se ejecute en él.
- Contenedores de Servlets fuera de proceso. Es una combinación de plugin para el servidor y una implementación del contenedor Java que se ejecuta es una máquina virtual de Java externa.

2.3.7 JavaMail

Adicionalmente al API Estándar de Java se puede encontrar esta expansión que permite el envío y recepción de correos electrónicos.

Este paquete es usado para crear programas del tipo MUA (Mail User Agent), cuya principal función es conectar con los programas MTA (Mail Tranfer Agent) para el envío y recepción del correo.

Permite gestionar cualquier tipo de correo a través de cualquier tipo de protocolo (POP, IMAP, SMTP, MIME…), pudiendo crear mensajes de texto plano o HTML, además de añadir imágenes, archivos adjuntos, etc.

2.3.8 IText

IText es una librería de Java con la que poder generar documentos PDF de forma dinámica, tanto en aplicaciones de escritorio como aplicaciones web.

Algunas de las posibilidades que ofrece son:

- Generar dinámicamente documentos usando archivos XML o Bases de datos.
- Manipular documentos ya creados.
- Agregar índices, números de página, marcas de agua…

2.4 Comparativa de las tecnologías disponibles

Actualmente, existe un gran abanico de posibilidades a la hora de afrontar un desarrollo Web. Todas ellas pretenden proporcionar características avanzadas a las páginas Web con el fin de cubrir las amplias necesidades de los usuarios, basándose en el esquema cliente-servidor.

La navegación Web ha evolucionado a pasos agigantados en los últimos años, haciendo que los diseñadores y programadores requieran herramientas más potentes, dejando a un lado los diseños estáticos basados en HTML.

El resultado de esta evolución es la gran cantidad de herramientas que se pueden encontrar para crear páginas muy potentes y dinámicas (CGI, JSP, PHP, ASP…)

Breve Descripción de cada tecnología.

- **CGI**: Mediante el uso de los Common Gateway Interface, los desarrolladores pudieron acceder a un mundo nuevo donde poder crear aplicaciones del lado del servidor y acceder a ellas a través de un navegador. Supuso un gran avance, pero su problema era su rendimiento, era demasiado bajo ya que cada acceso requería un nuevo proceso en el servidor.

- **PHP**: Es un lenguaje desarrollado por Rasmus Lerdorf en 1984 como un CGI escrito en C, permitiendo la interpretación de número limitados de comandos. Es de libre distribución, y desde su versión 4 pasa a ser un lenguaje interpretado, bastante rápido. Es sencillo, sintaxis cómoda, pudiéndose instalar en servidores gratuitos y asociándose con MySQL proporciona una gran potencia a la hora de desarrollar páginas Web. Como principal inconveniente tiene varios "Bugs" de seguridad.
- **ASP**: En 1996 Microsoft lanza ASP, con sintaxis y funcionamiento parecido a PHP, ofreciendo la ventaja de utilizar el concepto de objetos COM. Su principal problema es que solo permite su uso en plataformas Microsoft, limitando su portabilidad.
- **JSP**: Usa código HTML junto con código Java, siendo parecido a los lenguajes anteriores, separando claramente qué es contenido y qué es presentación. Reusa componentes basados en JavaBeans, y permite el uso de XML en los scripts.
- **Servltes**: Módulos de Java que se ejecutan en el servidor o contenedor JEE, expandiendo así las capacidades de los servidores Web. Se dice de ellos que son la siguiente etapa de los CGI, ya que su función más típica es generar páginas Web dinámicas a partir de los parámetros de la petición que envíe el navegador.
  - Los CGI fueron los primeros en aparecer, siendo así los más antiguos y desfasados.
  - JSP, ASP y PHP, son lenguajes enfrentados entre sí.
  - PHP es gratuito, fácil y rápido. Está continuamente en revisión y actualización. Es de código abierto y portable.
  - ASP. Ni es portable, ni de código abierto, ni gratis. Tiene soporte de Microsoft.
  - JSP. Portable, no es de código abierto, gratis y muy seguro.

2.4.1 Comparativa JSP y ASP

- JSP es portable, por lo que se puede ejecutar en cualquier sistema operativo y servidor Web, al contrario de ASP, que solo permite su ejecución en sistemas Microsoft y servidores muy concretos.
- JSP cuenta con una extensa API y comunidad Java. ASP tan solo cuenta con Microsoft para sus desarrollos.

- JSP Y ASP utilizan una combinación de TAGS y SCRIPTS para dar dinamismo a las páginas.
- JSP es independiente de la plataforma, los componentes son reutilizables tanto en Unix como Windows por ejemplo. ASP no lo permite.
- JSP utiliza como lenguaje Script Java, mientras que ASP utiliza VBScript o JScript. Java es mucho más seguro, fácil y potente que el resto de los lenguajes Script.
- Si lo que se desea es diseñar son pequeñas aplicaciones, los lenguajes Script son una posible alternativa, sin embargo, para aplicaciones grandes y bien estructuradas, la mejor opción vuelve a ser Java.

El uso de JSP se hace bastante claro respecto a ASP, siendo las dos ventajas más claras el uso de Java para la parte dinámica, y la portabilidad hacia otros sistemas operativos y servidores.

2.4.2 Comparativa PHP y JSP

- Tanto JSP como PHP son portables.
- PHP es de código abierto, mientras que JSP no.
- JSP permite crear nuevos TAGS mientras que PHP al igual que ASP no lo permite.
- Java sigue siendo mucho más completo y estructurado que cualquier otro lenguaje Script.
- La orientación a objetos en PHP aún no es tan sólida como en Java.

Decantarse por uno de estos dos lenguajes es algo más complicado, ya que no existen tantas diferencias y las ventajas de uno sobre otro no marcan una clara prioridad, sin embargo, Java es un lenguaje robusto, potente, bien estructurado, y de fácil comprensión, por lo que sería un posible motivo para decantarse por JSP.

Finalmente, la elección elegida para el desarrollo del proyecto ha sido:

- **Java**. Se trata de un lenguaje de programación multiplataforma, permite ser ejecutado en cualquier máquina, tan solo se debe tener la JVM. Es un lenguaje robusto y sencillo, con la ventaja de que se conocía previamente al inicio del

proyecto, por lo que el aprendizaje fue muy corto en este aspecto. Es más fiable que otros lenguajes, y permite la gestión de excepciones de una manera sencilla, labor que en este proyecto ha tenido que ser tratada de una maneara exhaustiva.

- **Jsp**. La utilización de Jsp viene en gran parte originada por la elección de Java. Se trata de un lenguaje compuesto de HTML/XML y scripts de Java, integrándose perfectamente con este último. En el proyecto, ha sido fundamental para la representación de los datos y su recogida. Al igual que Java, es portable e independiente de la plataforma, y tiene un API mucho mayor que cualquier otro lenguaje posible a utilizar en este aspecto.

- **MySQL**. Es un sistema de gestión de base de datos relacional muy rápido, multihilo, multiusuario y robusto. Permite altas cargas de datos, y es totalmente gratuito. Existen varias API que permiten acceder a las bases de datos desarrolladas en MySQL.

3 Metodología

Para la realización de esta sección, se han utilizado los estándares IEEE-1058 y IEEE-1074 como guía, sin ceñirse con exactitud a la metodología de cada uno.

3.1 Modelo de Ciclo de vida Software

Todo proyecto de ingeniería tiene unos fines ligados a la obtención de un producto o servicio que es necesario generar a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, necesario para continuar hacia el producto final y facilitar la gestión del proyecto. Al conjunto de fases empleadas se le denomina "ciclo de vida".

El ciclo de vida va unido al desarrollo del software, comprendiendo una seria de etapas que comprenden todas las actividades, desde el primer momento en el que se decide crear un nuevo producto, hasta el momento en el que se dispone a disposición de los usuarios.

Según la normativo ISO 12207 – 1, el ciclo de vida software se define como "Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso".

3.1.1 Objetivos

La definición de un ciclo de vida facilita:

- El control sobre los tiempos necesarios para aplicar recursos de todo tipo
- Control del trabajo subcontratado
- Control de calidad, siempre que la separación entre fases se haga corresponder con puntos en los que esta deba verificarse

Para el desarrollo del proyecto, se ha escogido el ciclo de vida "en cascada".

3.1.2 Elementos de un ciclo de vida

Un ciclo de vida esta compuesto por fases sucesivas compuestas a su vez por tareas planificables. Según el modelo de ciclo de vida, se puede componer también de bucles de realimentación, de manera que una misma fase pueda ser ejecutada más de una vez a lo largo del proyecto.

Para un adecuado control de la progresión del proyecto, es necesario especificar con precisión los productos intermedios que resultan de cada fase. Estos productos marcan los hitos entre las fases.

- **Fases**. Una fase es un conjunto de actividades relacionadas con un objetivo en el desarrollo del proyecto. Se construye agrupando tareas que pueden compartir un tramo determinado del tiempo de vida de un proyecto. La agrupación temporal de tareas impone requisitos temporales correspondientes a la asignación de recursos.

  Cada fase viene definida por un conjunto de elementos observables externamente, como son las actividades con las que se relación, los datos de entrada, los datos de salir y la estructura interna de la fase.

- **Entregables**. Son los productos intermedios que generan las fases. Pueden ser materiales o inmateriales. Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos. Cada una de estas evaluaciones puede servir para la toma de decisiones a lo largo del desarrollo del proyecto.

3.1.3 Modelo Ciclo de vida software en Cascada

En el modelo de ciclo de vida en Cascada, el proceso de desarrollo de software es disciplinado y planeado, en el que la implementación debe posponerse hasta que los objetivos se hayan comprendido.

Se distinguen principalmente las siguientes fases:

- **Análisis**. Se analizan las necesidades de los usuarios finales, con el fin de determinar los objetivos a cubrir. En esta fase se crea una memoria llamada SRD (Documento de Especificación de Requisitos), que contiene la especificación completa de lo que el sistema debe hacer sin entrar en detalles.

- **Diseño**. Se descompone y organiza el sistema en elementos que puedan elaborarse por separada, con el fin de aprovechar las ventajas del equipo de desarrollo. Se crea el documento SDD (Documento de Diseño de Software), que contiene la descripción de la estructura global del sistema y detalla qué debe hacer cada una de sus partes, así como la combinación con otras.

- **Codificación**. Básicamente es la fase de programación. Se desarrolla el código fuente, haciendo uso de pruebas y ensayos para corregir posibles errores.

- **Integración**. Se ensamblan los elementos ya programadas para componer el sistema, comprobando el buen funcionamiento de este antes de pasar a su explotación.

- **Mantenimiento**. Fase destinada a corregir errores o introducir mejoras a lo largo de la explotación del sistema. Se recoge en el Documento de Cambios.

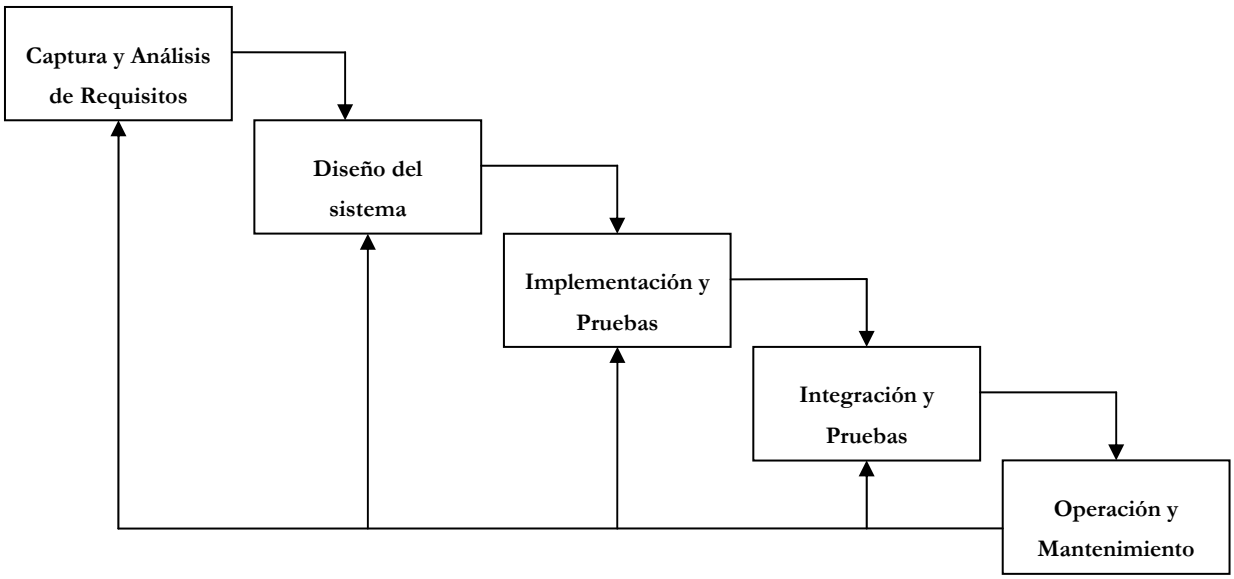

Ilustración 3-1 Ciclo de vida en cascada

Es un modelo descendente, que exige para pasar a la siguiente fase exige la conclusión de la fase anterior correctamente. La salida de una fase es la entrada de la siguiente.

3.2 Plan de proyecto

3.2.1 Resumen del proyecto

3.2.1.1 Propósito, alcance y objetivos

El propósito del proyecto es realizar una aplicación que permita la gestión de los voluntarios y asociaciones que hayan contratado a través de la empresa alguno de los seguros ofrecidos por esta, facilitando la generación de los certificados y correos pertinentes.

A su vez, la empresa desea en un futuro desarrollar otra aplicación con grandes similitudes a esta para gestionar seguros de aviación. Para ello, el programa será lo más genérico posible para permitir los posibles proyectos futuros.

La aplicación tratará las siguientes necesidades:

- Gestión de las asociaciones. Debe gestionar todas las asociaciones contratantes de seguros a través de la empresa. Se podrán realizar las siguientes acciones:
    - Dar de alta una nueva asociación. En un primer lugar se introducirán los datos disponibles de la asociación. Una vez que esta valide la información, se procederá a dar de alta a la asociación en el sistema.
    - Modificar los datos de una asociación. Se podrá modificar cualquier dato de la asociación, siempre y cuando esta este de acuerda.
    - Dar de baja una asociación. A petición de la asociación, se podrá realizar su baja en el sistema.
    - Consultar datos de las asociaciones. Se tendrá acceso a cualquier dato de las asociaciones.

- Gestión de voluntarios. Permitirá la gestión de los voluntarios pertenecientes a las asociaciones previamente dadas de alta.
    - Dar de alta voluntarios. Se facilitará el alta y la creación de los certificados individuales de cada voluntario.
    - Modificar información de los voluntarios. A petición de la asociación, se podrá modificar cualquier campo de los voluntarios.
    - Dar de baja voluntarios. Los voluntarios podrán ser eliminados en cualquier momento del sistema.
    - Establecer fecha de vencimiento del seguro, fecha de baja. Cada voluntario tiene su fecha de baja o vencimiento del seguro contratado, teniendo que ser esta establecida por las asociaciones.
    - Consultar datos de los voluntarios. Acceso total a los datos de los voluntarios pertenecientes a cada asociación.

- El alcance del proyecto agrupa las siguientes fases:
    - Planificación y requisitos
    - Diseño
    - Implementación
    - Integración y pruebas

La aplicación dispone de una página de ayuda donde poder consultar las acciones disponibles y el modo de ejecución de cada una de ellas.

Una vez entregada la aplicación, se deberá instalar en el servidor para su completo funcionamiento, y habiendo comprobado que no genera ningún tipo de error durante un tiempo determinado, se evitará el futuro mantenimiento de esta.

3.2.1.2 Restricciones y supuestos

Se determinan las siguientes restricciones y supuestos respecto a los diferentes campos, siendo los siguientes:

Calendario:

Como fecha límite se establece diciembre del mismo año, mes en el que vencen los seguros contratados y se deben realizar las siguientes pólizas.

Presupuesto:

Se acuerda un presupuesto máximo de 2000 euros para la realización de todas las etapas del proyecto.

Recursos:

No se impone ninguna restricción sobre los recursos a utilizar.

Software propio reutilizado:

El proyecto se realiza desde cero, por lo que no se utiliza software anterior.

Tecnologías a contratar:

No es necesario contratar ninguna, puesto que todas las utilizadas son software libre. Se pueden adquirir a través de Internet, siendo las descargas y su posterior instalación gratuitas.

La aplicación creada tendrá como objetivo poder ser reutilizada para la creación de otra aplicación sobre seguros de aviación.

Deberá ser instalada en un servidor que permita la utilización de JSP y MySQL.

En relación al software y hardware necesario por parte de la empresa para utilizar la aplicación, con una conexión a Internet y un navegador es suficiente.

3.2.1.3 Entregables del proyecto

El proyecto está dirigido a un cliente final, el cual no nos pide ningún tipo de documentación, tan solo que la aplicación funcione y un manual de usuario para poder usarla sin ningún tipo de problemas. Por lo tanto, el resto de documentación generada será guardada para reutilizarla en futuros proyectos.

Los entregables generados serán los siguientes:

- Código fuente
- Código ejecutable
- Base de datos
- Software de integración
- Paquete del software
- Información sobre los métodos de prueba
- Datos de prueba
- Software de prueba
- Análisis funcional
- Análisis de diseño
- Documentación del manejo de la aplicación y del funcionamiento

3.2.1.4 Horario y presupuesto

La planificación del proyecto se basa en las tareas que se obtendrán del estándar IEEE 1074. Dicho estándar proporcionará las tareas a realizar, las cuales deberán realizarse en el momento adecuado para cumplir con el calendario que se obtenga tras realizar la planificación.

A cada tarea se le asignaran unos recursos para su correcto desarrollo, los cuales habrán sido estimados en el plan de proyecto, así como el presupuesto del proyecto, el staff necesario para cada fase y la duración de las mismas.

Mirar anexos:

Anexo I: Diagrama Gantt

Apéndices A, B y C (RBS, WBS, PBS): RBS, WBS y PBS

3.3 Referencias

La lista de documentos que se han utilizado para el desarrollo de la aplicación ha sido la siguiente:

- IEEE1074-2006
- IEEE1058-1998

El resto de documentos esta listado en el capitulo Bibliografía.

3.4 Definiciones

GANTT → Gráfico ilustrativo sobre la planificación de tareas y las relaciones de precedencia (orden) entre las mismas.

HW → Hardware.

IEEE 1058 → Estándar para la realización del Plan de Proyecto Software.

IEEE 1074 → Estándar para el desarrollo del Modelo de Procesos Software.

PERT → Gráfico ilustrativo para realizar un seguimiento del proyecto y determinar el camino crítico.

RBS → Resource Breakdown Structure ⇨ Estructuración de recursos

SW → Software.

WBS → Work Breakdown Structure ⇨ Estructuración de tareas

3.5 Organización del proyecto

3.5.1 Interfaces externas

En este proyecto no hay relación con otras entidades (proveedores, contratistas). Por lo tanto no hay ninguna descripción de las relaciones entre el proyecto y entidades externas. La única organización con la que se interactúa es con el cliente, del que se extraerán los datos necesarios para realizar el proyecto, así como de los datos de la información que ellos manejan, para poder procesarla y traspasarla al modelo informático. Al final de cada fase, se comprobará por parte del responsable de gestión de calidad, que se están cumpliendo todos los planes definidos para el proyecto.

3.5.2 Estructura interna

Para este proyecto se necesitan las siguientes entidades para su correcto desarrollo. Un jefe de proyecto, que estará a cargo de los analistas, programadores y diseñadores necesarios, en este caso tan solo uno. Un jefe de calidad, responsable del encargado de pruebas y del encargado de aseguramiento de calidad, en esta caso, la misma persona.

La estructura queda de la siguiente manera:

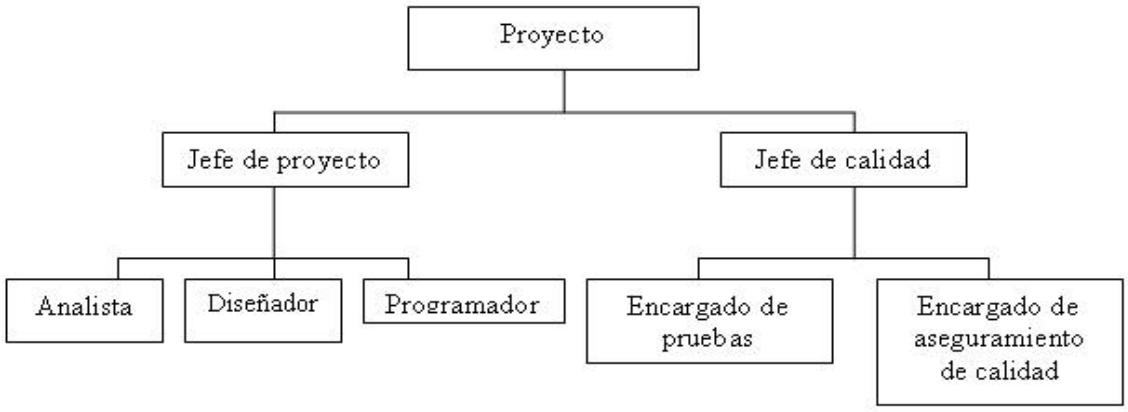

Ilustración 3-2 Estructura interna

3.5.3 Responsabilidades y roles

El equipo de proyecto es el siguiente:

- **Jefe de Proyecto**: Fernando Paniagua.
- **Jefe de calidad**: David Hernández.
- **Analista**: David Hernández.
- **Programador**: David Hernández.
- **Diseñador**: David Hernández.
- **Encargado de pruebas**: David Hernández.
- **Encargado de aseguramiento de calidad**: David Hernández.

A continuación se muestra la lista de actividades junto con los componentes o áreas del proyecto para determinar quien realiza cada una.

	Área de Calidad	Jefe de Proyecto	Analista	Diseño y Programación
1.1.1				
1.1.2				
1.1.3				
1.2.7				

1.3.2				
2.1.1				
2.1.2				
2.1.4				
2.2.1				
2.2.2				
2.2.3				
2.3.1				
2.3.2				
2.3.3				
2.3.4				
3.1.1				
3.1.2				
3.1.3				
3.2.1				
3.2.2				
3.2.3				
3.2.4				
3.3.1				
3.3.2				
3.3.3				
4.1.1				
4.1.2				
4.1.3				
5.1.1				
5.1.4				
5.1.5				
5.1.6				
5.1.7				
5.3.1				
5.3.2				
5.4.1				
5.4.2				
5.4.3				

Tabla 1 Lista de actividades

3.6 Planes de proceso de gestión

3.6.1 Plan de estimación

Para la estimación, se ha utilizado el diagrama Gantt (Mirar Anexo I). Este diagrama permite conocer las diferentes tareas a realizar a lo largo del tiempo total determinado. También, permite conocer los costes de cada tarea y la persona encargada en desarrollarla.

La revisión del diagrama será constante durante el desarrollo del proyecto.

3.6.2 Plan de plantilla

Mirar Apendice WBS.

3.6.3 Plan de compra de recursos

Para el desarrollo del proyecto se necesita el siguiente hardware y software.

3.6.3.1 Software

- Windows XP Profesional
- J2SE
- MySQL
- Microsoft Office XP
- Dreamweaver
- Netbeans
- Apache Tomcat

Ninguna de ellas se debe adquirir, por lo que no supone un gasto para el cliente final, al igual que el Hardware utilizado.

3.6.3.2 Hardware

Ordenador de sobremesa Intel Core Duo 2.00 GHz., 1 Gb memoria RAM, 200 Gigas Disco Duro.

Para la instalación de la aplicación, se requerirá un servidor que soporte las tecnologías JSP, Servlets y MySQL. Si el cliente dispone uno con estas características, la instalación se realizará sobre este, si por el contrario el servidor es externo al cliente y hay que contratarlo, los gastos correrán a cargo del cliente.

Si tampoco dispone de un dominio Web, se contratará junto con el servidor.

Se estima que estos costes se situan en torno a los 100 – 150 euros, dependiendo del tipo de alojamiento, cantidad de cuentas de correo requeridas, capacidad de alojamiento, etc.

3.6.4 Plan aprendizaje personal

La plantilla consta de dos personas, las cuales ocupan varios puestos diferentes a la vez. El jefe de proyecto y calidad conoce todas las tecnologías que se van a utilizar, por lo que no necesita ninguna formación adicional. Sin embargo, el otro componente del proyecto, encargado de la programación, pruebas, análisis, etc., no dispone de la formación adecuada, por lo que deberá aprender de forma autodidacta a utilizar las siguientes tecnologías:

- Servltes
- JSP
- MySQL
- Paquete JavaMail
- Paquete IText

Se da por hecho que domina el resto de software a utilizar.

Complementariamente también aprenderá nociones básicas sobre seguros, pólizas, voluntariado, etc., con el fin de orientar la aplicación de una forma más adecuada hacia el cliente final.

Esta formación servirá de gran ayuda para posibles proyectos futuros en los que se vuelvan a utilizar estas tecnologías.

3.6.5 Plan de Trabajo

3.6.5.1 Actividades

En este punto se especifican las actividades que se llevarán a cabo durante todo el proyecto, indicando de este modo el plan de trabajo que se intentará seguir para el desarrollo de la aplicación.

Para ello, se ha requerido la construcción de la Estructura de Desglose de Tareas (WBS) (Mirar Apéndice B (WBS)).

El WBS nos da una indicación de las tareas que a ejecutar y su posición dentro del ciclo de vida.

Mirar anexos:

Anexo I: Diagrama Gantt

Apéndice B (WBS): WBS

3.6.5.2 Planificación

La asignación de horarios se ha realizado a través de la aplicación MS Project, mostrando las actividades requeridas para el desarrollo del proyecto junto con su correspondiente fecha y persona al cargo.

Se modificarían fechas si fuese necesario, intentando que el impacto en el proyecto fuera el menor posible.

Mirar anexo:

Anexo I: Diagrama Gantt

3.6.5.3 Adquisición de recursos

El plan para adquisición de recursos esta especificado en el ***Apéndice A (RBS)***. No se adquirirá en este proyecto ni servicio de transporte, instalaciones, servicios administrativos y de limpieza.

Mirar anexo:

Apéndice A (RBS): RBS

3.6.5.4 Asignación de presupuesto

A continuación se muestra un desglose del coste de cada etapa.

- Actividades previas 864 Euros

- Gestión 1764 Euros

- Análisis 4401 Euros

- Diseño 5676 Euros

- Formación personal 194 Euros

- Implementación 3826 Euros

- Pruebas 1644 Euros

- Instalación Software 280 Euros

- Aceptación del Software 320 Euros

- Distribución documentación 320 Euros

Total presupuesto: 19288 Euros

De una forma más detallada y precisa, se muestra el coste en la siguiente tabla.

Tarea	Coste (€)
Crear código ejecutable	2024
Identificar y analizar requisitos funcionales	1560
Desarrollar requisitos	1280
Ejecutar pruebas	1104
Diseñar base de datos	992
Diseñar interfaces	992
Diseño detallado	992
Identificar ideas o necesidades	741.82
Planificar gestión proyecto	640
Priorizar e integrar requisitos	620
Identificar requisitos	603.08
Formular posibles propuestas	560
Documentar la implementación	530.40
Implementar manual de usuario	530.40
Ejecutar estimación	521.14
Concretar necesidades cliente	520
Diseñar arquitectura	520
Descomponer requisitos del sistema	480
Integración de partes del sistema	440
Aceptación del software	320
Producir y distribuir documentación	320
Investigación	288
Elegir ciclo de vida	288
Crear proceso ciclo de vida	288

Búsqueda de integración software	285.60
Búsqueda de software de desarrollo	280
Resultado de pruebas	280
Instalación del software	280
Crear datos de prueba	260
Definir y analizar requisitos SW	240
Definir y analizar requisitos interfaz	240
Formación del personal	192
Analizar arquitectura del sistema	60
	TOTAL
	19288.44

Tabla 2 Presupuesto detallado de la aplicación

3.6.6 Plan de Control

3.6.6.1 Plan de control de requisitos

Se tendrá que llevar a cabo un control de los requisitos durante todo el desarrollo del proyecto, con el fin de conocer si se están cumpliendo los objetivos y necesidades marcados en un principio.

El control será llevado a cabo por los componentes del equipo de trabajo, ya que es posible que el desarrollo puede tener modificaciones en alguna de las etapas, añadiendo o eliminando actividades para un mejor desarrollo del proyecto. Las modificaciones no afectan a las restricciones impuestas anteriormente el diseño previo.

3.6.6.2 Plan de control de planificación

Con el fin de controlar todos los tiempos, tanto de entrega como de plantilla, se creará un plan de control de cronograma que permitirá gestionar de una manera eficiente y sencilla las fechas y horarios previstos durante todo el desarrollo.

Así pues, se deberá seguir rigurosamente el calendario que proporciona MS Project mediante el diagrama GANTT (***Mirar Anexo I GANTT***), mostrando claramente las fechas y días que se debe utilizar para cada actividad del proyecto. Es posible que se realicen modificaciones de fechas y entregas, así que este diagrama muestra una estimación previa del calendario, pero a lo largo del desarrollo puede que se vuelva a tener que estimar.

3.6.6.3 Plan de control del presupuesto

No aplica.

3.6.6.4 Plan de control de calidad

No aplica.

3.6.6.5 Plan de informes

No aplica.

3.6.6.6 Plan de recolección de métricas

No aplica.

3.6.7 Plan de gestión de riesgo

No aplica.

3.6.8 Plan de cierre

Al finalizar el proyecto, se propondrá a la empresa contratante un servicio de mantenimiento con el fin de revisar el buen funcionamiento de la aplicación, así como posibles modificaciones que se requieran.

3.7 Planes de proceso técnico

3.7.1 Modelo de proceso

El modelo en el que se ha basado el desarrollo ha sido el IEEE 1074, el cual determinará el ciclo de vida en cascada elegido para el proyecto.

Consta de 4 fases:

Captura y Análisis de Requisitos:

- Se planifica el proyecto, abarcando el desarrollo software y todas las tareas de gestión del software.
- Se identifican las ideas y necesidades, se formulan aproximaciones al potencial del producto, se estudia la viabilidad.
- Se concretan los requisitos sobre el sistema mediante reuniones con el usuario
- Se establecen los requisitos *hardware* y *software* necesarios

Diseño del Sistema

- Se definen los requisitos software por parte del usuario, tales como requisitos funcionales, de interfaces… y priorizarlos.
- Se define el organigrama del funcionamiento de la aplicación.
- Se define el medio y forma del enlace entre el usuario y la aplicación.
- Se diseña un plan de pruebas para el buen funcionamiento de la aplicación.

Implementación

- Se codifica la aplicación.
- Se realiza la incorporación de todas las partes y módulos de la aplicación al sistema.
- Se crean los informes necesarios para el usuario.

Integración y Pruebas

- Se lleva acabo la instalación del software.
- Se realizan las pruebas definitivas para el correcto funcionamiento de la aplicación, y el usuario da el visto bueno.

- Se ofrece formación para que los usuarios conozcan la aplicación y su perfecto manejo.

3.7.2 Métodos, herramientas y técnicas

Las herramientas que se han utilizado para confeccionar el desarrollo de la planificación del proyecto son las siguientes:

- Microsoft Project.
- Microsoft Word.
- Microsoft PowerPoint.
- NetBeans
- Lenguajes de programación

Los lenguajes de programación usados son:

- Java
- MySQL
- JSP

3.7.3 Plan de infraestructura

Hardware utilizado:

- Pentium Quad Core 2.40 GHz, 1 Gb RAM.
- Red ADSL 2 MB.

Software utilizado:

- Windows XP
- NetBeans
- Dreamweaver

Sistema anfitrión:

- - Pentium 4 1.60 GHz, 512 MB RAM.
- - Windows XP
- - Red ADSL

3.7.4 Plan de aceptación del producto

Se debe dar al cliente la aplicación dentro del plazo estimado. Una vez entregada, se le facilitarán las pruebas si fuese necesario, y habrá un tiempo de prueba para ver el correcto funcionamiento de la aplicación en el día a día.

Si el cliente esta conforme con la aplicación durante este tiempo, la venta se dará por cerrada, y comenzará el periodo de mantenimiento si así lo desean.

3.8 Plan de proceso

3.8.1 Plan de gestión de la configuración

Se utilizará la herramienta CVS (Concurrent Versions System) que es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.

El modo en el que se nombran las versiones será:

v.XX.YY.ZZZZ, siendo:

v: indicador de que es una versión

XX: el número de la versión

YY: el número de release

ZZZZ: el número de parche

Numerándose secuencialmente de la siguiente manera:

En el caso de introducir un nuevo parche: v.XX.YY.ZZZZ+1

Si se introduce una nueva release para solucionar cualquier anomalía o error del sitema: v.XX.YY+1.0000

Si se modifica la versión, añadiéndole nuevas funcionalidades, sería: v.XX+1.00.0000

Para la correcta utilización de ésta herramienta, se tendrán almacenados en una base de datos las versiones de los distintos ficheros que formen el proyecto, al acceder a un fichero, éste se quedará bloqueado (a no ser que varias personas tengan que trabajar con los mismos archivos) hasta que el usuario vuelva a subirlo y realice el correspondiente control de cambios.

3.8.2 Plan de validación y verificación

El propio CVS permite que varios usuarios puedan sacar copias del proyecto al mismo tiempo. Posteriormente, cuando actualizan sus modificaciones, el servidor trata de acoplar las diferentes versiones. Si esto falla, por ejemplo debido a que dos clientes tratan de cambiar la misma línea en un archivo en particular, entonces el servidor deniega la segunda actualización e informa al cliente sobre el conflicto, que el usuario deberá resolver manualmente. Si la operación de ingreso tiene éxito, entonces los números de versión de todos los archivos implicados se incrementan automáticamente, y el servidor CVS almacena información sobre la actualización, que incluye una descripción suministrada por el usuario, la fecha y el nombre del autor y sus archivos log.

Se realizará con cada versión su correspondiente control de errores, realizando varias pruebas introduciendo en el sistema todos los posibles valores, esperados o no, y forzando a la aplicación para comprobar su funcionamiento en situaciones límite.

3.8.3 Plan de documentación

Este plan hace referencia a toda la documentación del proyecto software y a las entidades responsables de proporcionar la información de entrada y de generar y repasar los distintos documentos.

Productos no entregables: no se entregan pero son necesarios para llevar a cabo la realización de la aplicación. Son los siguientes:

- Especificación de requisitos
- Documentación de la organización (WBS, RBS, PBS, matriz de trazabilidad)
- Documentación de la planificación (diagrama Gantt)
- Documentación del diseño
- Planes de prueba

Productos entregables*:* hacen referencia a toda la documentación creada para la instalación y el mantenimiento de la aplicación, así como la documentación de ayuda y de formación para los usuarios que vayan a utilizarla. Son los siguientes

- Código fuente
- Manual de usuario
- Sistema de ayuda
- Documentación de la configuración
- Guía de mantenimiento

3.8.4 Plan de garantía de calidad

La calidad de un proyecto es una de las actividades más importantes para determinar la entrega correcta del mismo, y se realiza a lo largo del ciclo de vida del proyecto. El propósito de este plan es determinar si un objeto es mejor, peor o igual que otro objeto de la misma especie. Para ello se atiende a dos conceptos:

- Tiene que satisfacer al cliente.
- Tiene que minimizar errores (aunque se debe tener en cuenta que en un proyecto software la existencia de errores nunca será cero).

La gestión de la calidad debe tener una concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente. En resumen, la gestión de la calidad debe tener tres hechos básicos:

- Plan de calidad => basado en los estándares.
- Reglas que han de cumplirse => son los requisitos de calidad.
- Aseguramiento de la calidad => llevado a cabo por el responsable de calidad.

Para asegurar la calidad de un proyecto se debe poder responder afirmativamente a una serie de preguntas sobre la aplicación, cada una de las cuales está relacionada con uno de los siguientes once factores:

- Según el punto de vista de la revisión:

- Facilidad de mantenimiento: El programa se puede arreglar con bastante facilidad. Tiene un valor medio.
- Facilidad de prueba: A la aplicación se le podrán pasar los programas de prueba con total normalidad. Tiene un valor alto.
- Flexibilidad: Gracias a la modularidad del programa, éste tiene una alta flexibilidad. Tiene un valor alto.

- Según el punto de vista de la transición:
  - Interoperabilidad: El programa no se puede comunicar con otros sistemas.
  - Portabilidad: Siempre y cuando se cumplan los requisitos mínimos de instalación de la aplicación, ésta se podrá utilizar en diferentes máquinas. Tiene un valor medio.
  - Reusabilidad: La aplicación se ha implementado de tal forma que garantiza su reusabilidad. Tiene un valor alto.

- Según el punto de vista de la operación:
  - Corrección: La aplicación software debe cumplir los requisitos estipulados, por tanto, deberá cumplir con la finalidad del producto. Tiene un valor alto.
  - Fiabilidad: Se intentará que el programa funcione de forma exacta todo el tiempo. Tiene un valor alto.
  - Eficiencia: Se intentará que la aplicación se ejecute sobre el hardware lo mejor posible. Tiene un valor alto.
  - Integridad: No es una de las principales finalidades. Tiene un valor bajo.
  - Facilidad de uso: La aplicación se podrá ejecutar de forma sencilla, para que al cliente le resulte lo más cómodo posible. Tiene un valor medio.

En resumen, el orden de prioridad de estos 10 factores para el plan de calidad de nuestro proyecto es el siguiente:

1. Corrección
2. Fiabilidad
3. Eficiencia
4. Facilidad de uso
5. Facilidad de mantenimiento
6. Facilidad de prueba
7. Flexibilidad
8. Reusabilidad
9. Portabilidad
10. Integridad

3.8.5 Revisiones y auditorías

No aplica.

3.8.6 Plan de resolución de problemas

No aplica.

3.8.7 Plan de gestión de subcontratación

No aplica.

3.8.8 Plan de mejora del proceso

No aplica.

3.8.9 Planes adicionales

No aplica

.

3.9 Apéndices

3.9.1 Apéndice A (RBS)

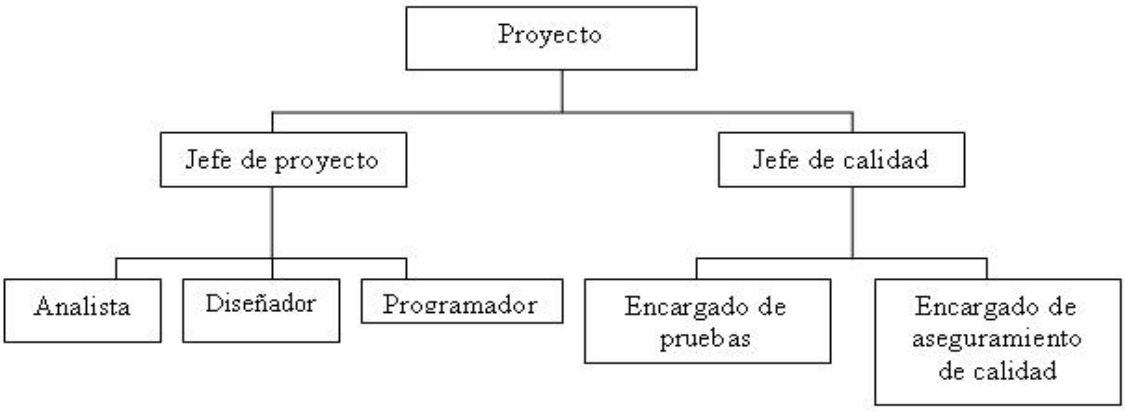

Ilustración 3-3 RBS

3.9.2 Apéndice B (WBS)

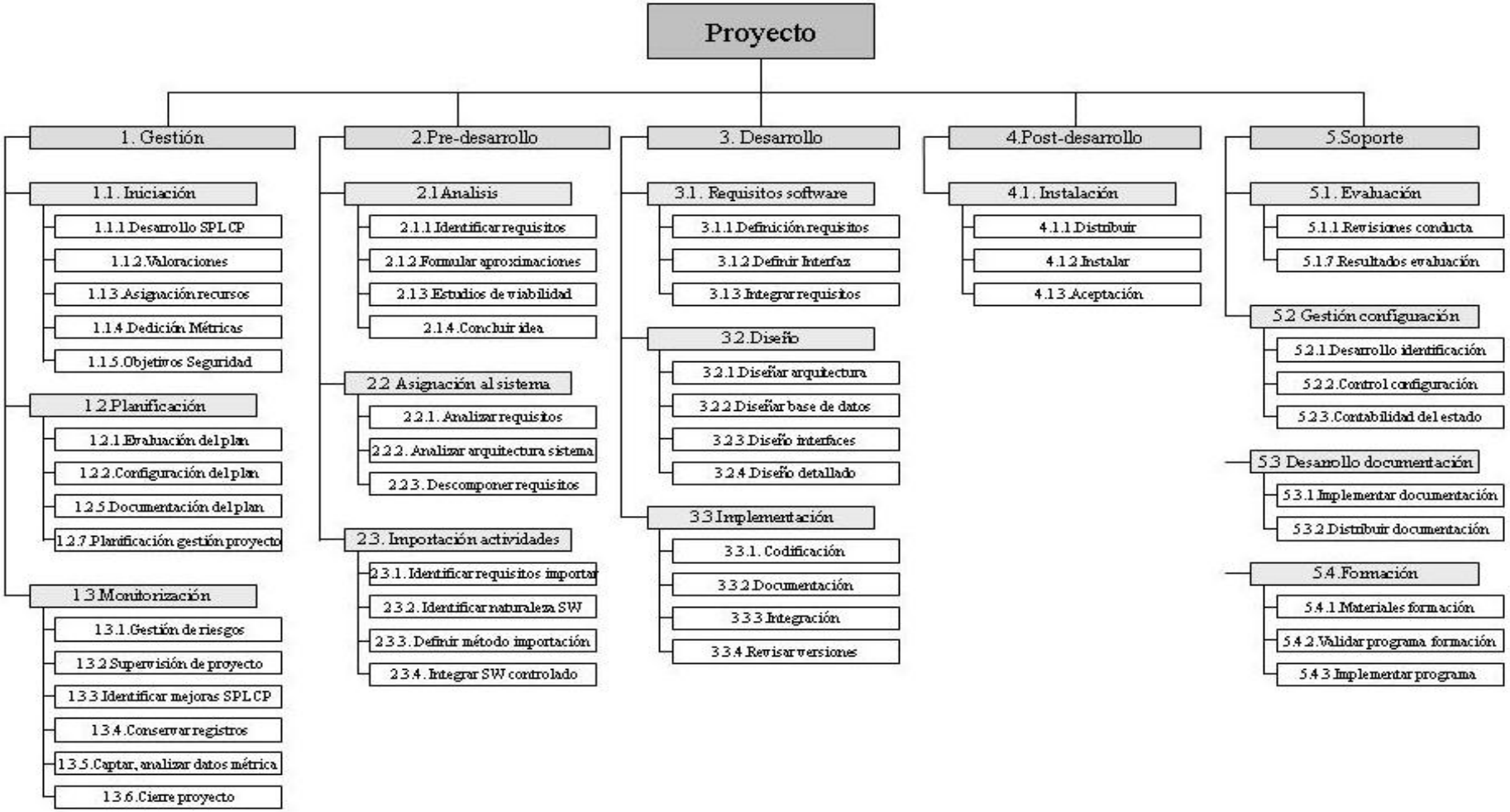

Ilustración 3-4 WBS

3.9.3 Apéndice C (PBS)

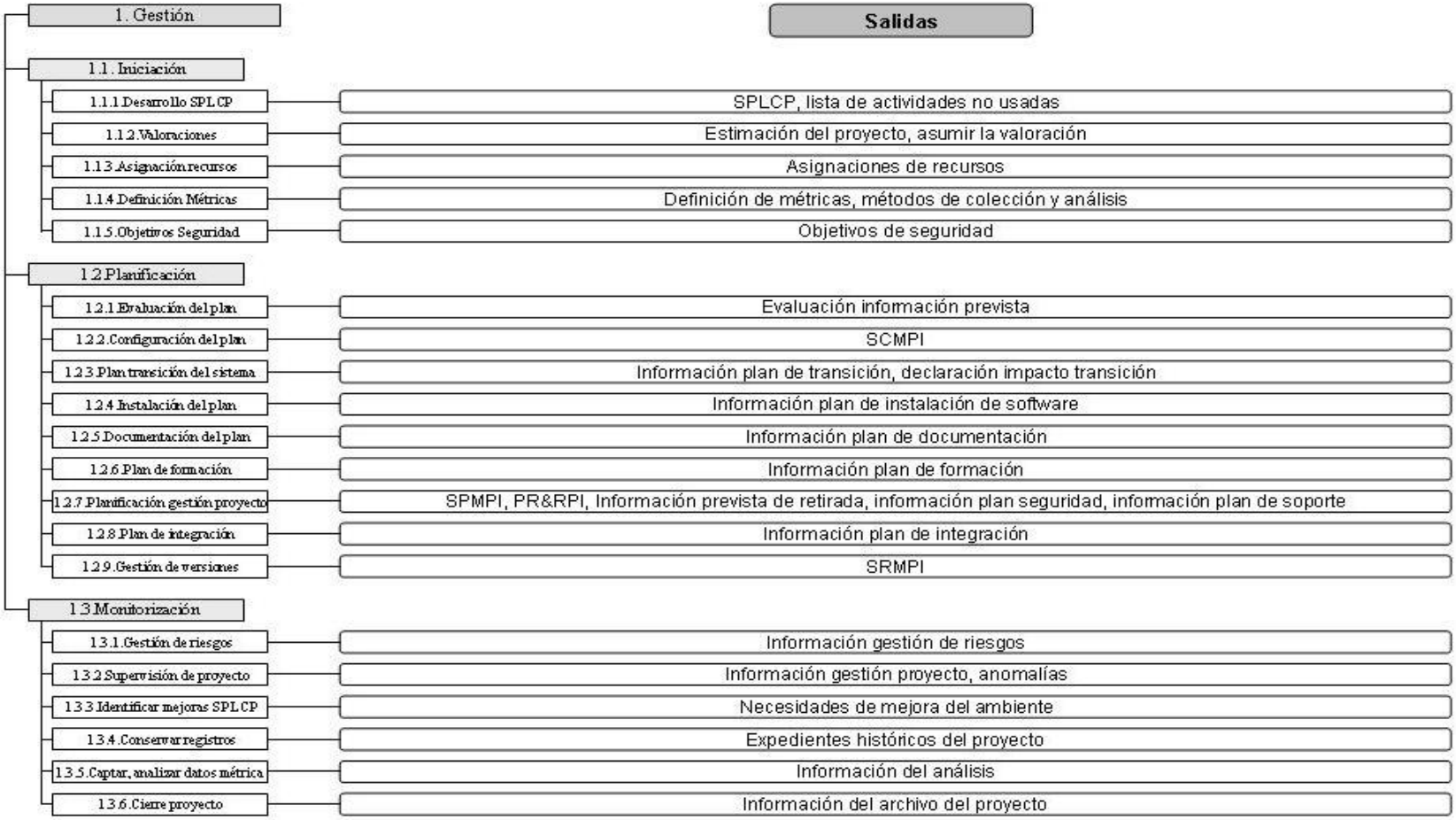

Ilustración 3-5 PBS Gestión

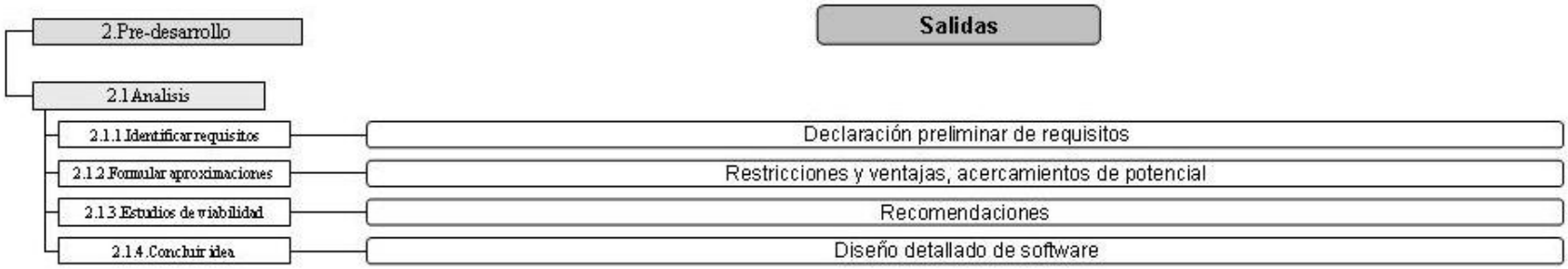

Ilustración 3-6 PBS Pre-desarrollo

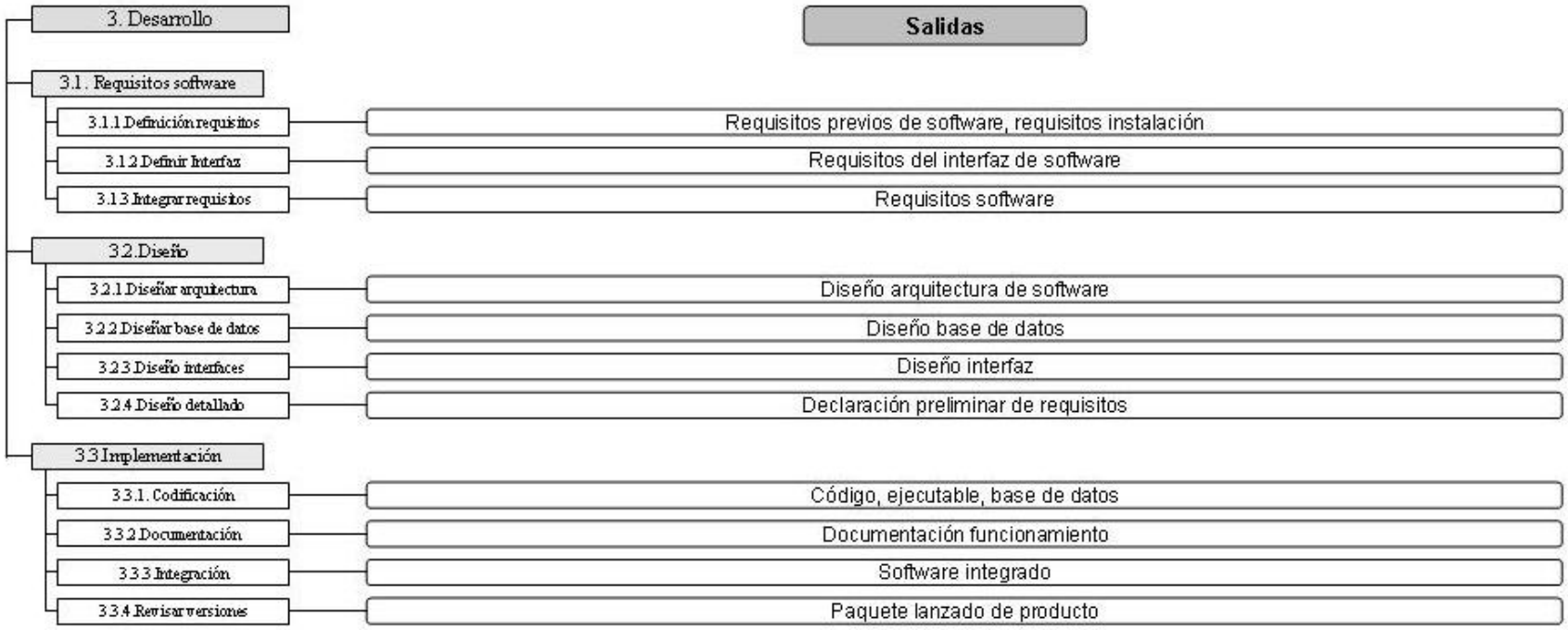

Ilustración 3-7 PBS Desarrollo

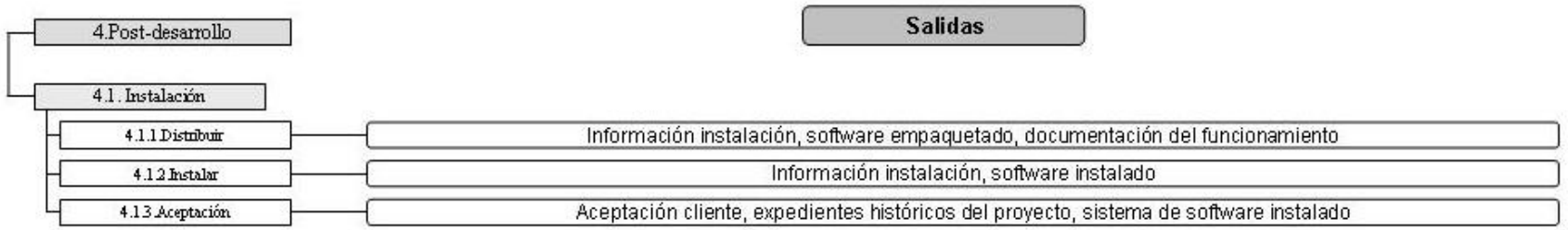

Ilustración 3-8 PBS Post-desarrollo

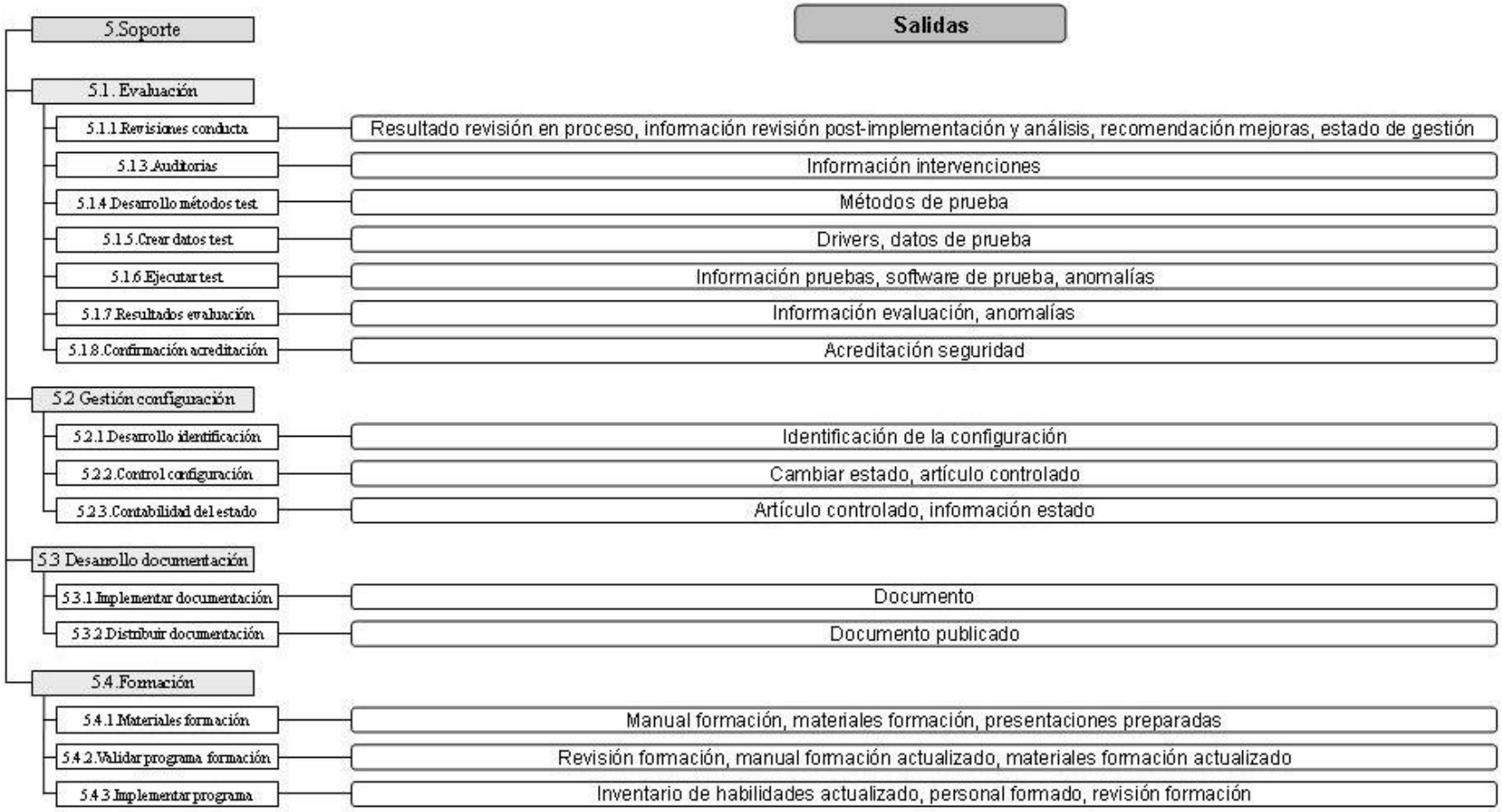

Ilustración 3-9 PBS Soporte

4 Análisis

4.1 Descripción del sistema

La aplicación consistirá en la gestión de voluntarios pertenecientes a distintas asociaciones.

Para ello, existirá la figura de administrador, en este caso, la empresa que solicita la aplicación. El administrador podrá realizar las siguientes tareas con el fin de facilitar y controlar la gestión de la información:

- Podrá dar de alta una asociación introduciendo tanto los datos corporativos como el domicilio de esta. La aplicación enviará un contrato que deberá ser firmado y rellenado por la asociación. Este, volverá a ser enviado al administrador, que será el encargado de validar los datos en la aplicación. Si todo el proceso ha sido correcto, el sistema enviará un correo a la asociación con los datos necesarios para ingresar en la aplicación.

- En el caso de que una asociación quiera darse de baja en el sistema, o bien caduque su fecha de efecto, el administrador se encargará de borrarla de la base de datos. Si la asociación es eliminada, también lo serán sus voluntarios.

- Si la asociación ha sido dada de alta con algún fallo en sus datos, el administrador podrá modificarlos si así se requiere. Toda modificación será notificada tanto al correo de la asociación como al correo de la persona de contacto dentro de esta. Si la asociación pierde la clave o desea que se le reenvíe, será el administrador el encargado de realizar este proceso.

- El administrador podrá dar de alta a voluntarios, aunque este no sea su cometido. Deberá rellenar los datos pertinentes y seleccionar la asociación a la que pertenecen. Este proceso generará un correo con los certificados que será enviado a la asociación. A su vez, se enviará otro correo para notificar tanto al administrador como a la sociedad, de que el alta se ha realizado correctamente.

- Si se desea realizar una carga masiva de voluntarios, el administrador será quien se ocupe de ello. Se deberá disponer de un archivo de texto con el formato correcto. Si hubiese algún fallo en el formato, se notificará por pantalla de los errores. De lo contrario, los voluntarios serán dados de alta en la asociación indicada. En este caso, no se generan los certificados, ya que o bien esos voluntarios ya han sido registrados en la sociedad con anterioridad, o bien, se les enviará los certificados por otro medio.

- En el caso de tener que dar de baja algún voluntario, la asociación deberá comunicárselo al administrador. También, y aunque no sea tarea del administrador, se podrá establece una fecha de baja para los voluntarios.

- Si algún voluntario contiene algún error en sus datos, el administrador será el encargado de modificarlos. Esta modificación no será notificada a la asociación.

- El administrador podrá consultar en cualquier momento las asociaciones y voluntarios disponibles en la base de datos. La búsqueda se realizará a través de un criterio de búsqueda definido por el administrador.

Las tareas a realizar por la asociación son:

- Modificar los datos de la asociación. Al igual que con el administrador, la asociación podrá modificar su datos. Toda modificación se notificará por correo tanto a la persona de contacto de la asociación, como al correo facilitado por la asociación.

- Dar de alta a voluntarios. Será la principal tarea de la asociación, dar de alta a los voluntarios sin tener que informar al administrador. Deberán rellenar los datos de los voluntarios, y si estos son correctos, se enviará un correo con los certificados correspondientes a cada uno de ellos al correo de la persona de contacto. También, se informará al administrador y a la sociedad de los voluntarios añadidos.

- Dar de baja a los voluntarios. La asociación podrá dar de baja a los voluntarios cuando lo requieran. También podrán establecer una fecha de baja para cualquiera de ellos.

- Podrán consultar la información de los voluntarios y generar un libro de registro con los voluntarios consultados. El libro de registro tendrá un formato adecuado para que tenga validez.

4.2 Requisitos de Usuario y Sistema

En esta sección se presentan los requisitos de usuario que deberán ser satisfechos por el sistema. Todos ellos son esenciales, por lo que se tendrá que poner especial énfasis en completarlos.

Identificador	**RUC-001**	**Versión**	1.0
Nombre	**Restricción de entrada a la aplicación**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	La aplicación deberá constar de un sistema de validación que a través de un nombre de usuario y una clave específica para cada usuario, permita el acceso a ella. Si alguna persona ajena al sistema pretende entrar, la aplicación deberá restringir el acceso informando por pantalla de la invalidez de sus datos. Igualmente, si un usuario intenta entrar y comete un error en la inserción de sus datos, la aplicación le informará por pantalla de su error.		

Tabla 3 Requisito 001

Identificador	RUC-002	Versión	1.0
Nombre	**Eliminación de los datos de voluntario**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	El sistema debe proceder a la eliminación de los voluntarios pertenecientes a una asociación cuando se elimine esta. Para ello, la aplicación contará con un sistema que automáticamente borre la información. Este proceso no debe generar ninguna notificación al administrador del sistema.		

Tabla 4 Requisito 002

Identificador	RUC-003	Versión	1.0
Nombre	**Notificación de modificación de datos**		
Fuente	Cliente		
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Prioridad	☐ Alta ☑ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	Puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Cuando tanto el administrador del sistema como el de una asociación modifiquen la información referente a una asociación, deberá ser notificado vía e-mail a los correos correspondiente. Se enviará un correo a la asociación para que compruebe si los datos modificados son correctos. Se enviará otro correo al administrador de la asociación, por si la modificación hubiese sido incorrecta.		

Tabla 5 Requisito 003

Identificador	RUC-004	Versión	1.0
Nombre	**Envío de certificados**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	Puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Cuando el administrador del sistema o de una asociación proceda a dar de alta a nuevos voluntarios, el sistema deberá generar los certificados correspondientes a cada uno de ellos. Una vez generados todos, se adjuntan a un correo que será enviado al administrador de la asociación. Puede ser que los certificados se modifiquen a lo largo del desarrollo.		

Tabla 6 Requisito 004

Identificador	RUC-005	Versión	1.0
Nombre	**Envío de clave**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Si se diese el caso de que algún administrador de asociación olvidase la clave de acceso al sistema, el administrador del sistema deberá poder reenviar la clave por correo. La notificación de la pérdida de la clave deberá ser notificada por un medio externo a la aplicación, como puede ser mediante un correo electrónico o una llamada telefónica.		

Tabla 7 Requisito 005

Identificador	RUC-006	Versión	1.0
Nombre	**Fechas correctas**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	La aplicación debe comprobar que las fechas introducidas sean correctas y cumplan el formato establecido. Gran parte de la documentación generada por el sistema depende de las fechas introducidas por los usuarios, por lo que hay que poner especial atención en este apartado. El formato a seguir será el siguiente: dd/mm/aaaa		

Tabla 8 Requisito 006

Identificador	RUC-007	Versión	1.0
Nombre	**Restricción de la modificación de voluntarios**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Si el administrador de la asociación desea modificar los datos de alguno de sus voluntarios, deberá ponerse en contacto con el administrador del sistema, que será el encargado de realizar los cambios pertinentes. Esta restricción evita posibles modificaciones indeseadas en los datos de los voluntarios. Las modificaciones de los voluntarios no serán notificadas a ningún usuario.		

Tabla 9 Requisito 007

Identificador	RUC-008	Versión	1.0
Nombre	**Carga masiva de datos**		
Fuente	Cliente		
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Prioridad	☐ Alta ☑ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Cabe la posibilidad que en el primer momento de puesta en marcha de la aplicación, se deba hacer una carga masiva de datos a través de ficheros. En este caso, la aplicación debe de tener un sistema que garantice la carga de los voluntarios a través de ficheros txt con el formato correcto, notificando aquellos errores que el fichero contenga antes de dar de alta ningún voluntario. Una vez realizada la carga masiva, esta opción permanecerá activa ya que es posible que alguna asociación quiera enviar un archivo con todos sus voluntarios y delegar en el administrador la tarea de introducir los voluntarios.		

Tabla 10 Requisito 008

Identificador	RUC-009	Versión	1.0
Nombre	**Baja de las asociaciones**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Las asociaciones no podrán darse de baja en la aplicación a través de su interfaz. Si desean darse de baja, deberán notificarlo a través de correo electrónico o cualquier otro medio al administrador del sistema, que una vez reciba la petición, proceda a eliminar la asociación de la base de datos.		

Tabla 11 Requisito 009

Identificador	RUC-010	Versión	1.0
Nombre	**Fechas de los voluntarios**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Al dar de alta a los voluntarios, las fechas serán generadas o introducidas dependiendo del usuario que realice el proceso. Si se trata del administrador del sistema, la fecha de alta será generada automáticamente, correspondiendo con la fecha de alta en el sistema, mientras que la fecha de inicio quedará vacía a la espera de rellenarse por la asociación. Si es el administrador de la asociación quien realiza el proceso, deberá introducir desde un primer momento la fecha de inicio, y al igual que antes, la fecha de alta corresponderá a la de alta en el sistema.		

Tabla 12 Requisito 010

Identificador	RUC-011	Versión	1.0
Nombre	**Fecha de baja de los voluntarios**		
Fuente	Cliente		
Necesidad	☐ Esencial ☐ Deseable ☑ Opcional		
Prioridad	☐ Alta ☐ Media ☑ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☐ Alta ☑ Media ☐ Baja		
Estabilidad	Puede sufrir modificaciones a lo largo del proyecto.		
Descripción	La fecha de baja de los voluntarios será establecida por el administrador de la asociación correspondiente. No es un proceso importante, ya que salvo rara vez, la fecha de baja corresponderá con la fecha de efecto de la que disponga la asociación más un año-		

Tabla 13 Requisito 011

Identificador	RUC-012	Versión	1.0
Nombre	**Búsqueda por campos**		
Fuente	Cliente		
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Prioridad	☐ Alta ☑ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☐ Alta ☑ Media ☐ Baja		
Estabilidad	Puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Las consultas deben poder buscar a través de parámetros incompletos. Si se introduce el principio de una palabra, se deben sacar todas las ocurrencias que contengan en su comienzo el parámetro a buscar. Esto se debe cumplir tanto para las búsqueda en asociaciones como en voluntarios Si se desea listar todos los voluntarios o asociaciones, deberán dejarse en blanco los parámetros de búsqueda.		

Tabla 14 Requisito 012

Identificador	RUC-013	Versión	1.0
Nombre	**Libro de registro**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	El sistema debe presentar un libro de registro con un formato establecido para que les sirva a las asociaciones como un libro de registro real. Se podrá realizar un libro de registro de unos voluntarios escogidos mediante una búsqueda previa.		

Tabla 15 Requisito 013

Identificador	RUC-014	Versión	1.0
Nombre	**Contrato autorización**		
Fuente	Cliente		
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Prioridad	☑ Alta ☐ Media ☐ Baja		
Claridad	☑ Alta ☐ Media ☐ Baja		
Verificabilidad	☑ Alta ☐ Media ☐ Baja		
Estabilidad	No puede sufrir modificaciones a lo largo del proyecto.		
Descripción	Al dar de alta una asociación, se debe enviar un contrato al correo de esta, con el fin de que lo devuelva firmado y confirmando el alta en la base de datos de la aplicación. La asociación no podrá usar el sistema hasta que envíe el contrato junto con los datos personales que se requieren. Una vez recibida dicha información, el administrador pasara a validarla y enviar la contraseña de acceso al sistema.		

Tabla 16 Requisito 014

5 Diseño

5.1 Arquitectura

La arquitectura de software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación, y tiene la responsabilidad de:

- Establecer los fundamentos para que analistas, programadores, diseñadores, etc., trabajen en una línea común que permita alcanzar los objetivos marcados.
- Proporcionar patrones y abstracciones coherentes que permitan guiar la construcción del software para un sistema de información.
- Seleccionar y diseñar la estructura idónea en base a los objetivos y restricciones.
- Definir los módulos principales.
- Definir las responsabilidades de cada modulo.
- Definir la interacción entre los módulos.

Proporciona así una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos a pasos posteriores del diseño.

La definición oficial de arquitectura de software según el estándar IEEE 1471-2000, es la siguiente: "*La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantan, y los principios que orientan su diseño y evolución*".

El objetivo principal de la arquitectura del software es aportar los elementos necesarios para ayudar a la toma de decisiones, además de aportar conceptos y el lenguaje común para permitir la comunicación entre los equipos que participen en el proyecto. Para ello, la arquitectura del software construye abstracciones en forme de diagramas.

No hay ningún estándar en cuanto a la forma y el lenguaje a utilizar en los diagramas, pero si existe un consenso en cuanto a la necesidad de organizar las abstracciones en vistas.

Existen al menos tres vistas fundamentales en cualquier arquitectura.

- Visión estática. Describe que componentes tiene la arquitectura.
- Visión funcional. Describe que hace cada componente.
- Visión dinámica. Describe el comportamiento y la relación que hay entre los componentes.

Arquitecturas más comunes

Monolítica. El software se estructura en grupos funcionales muy acotados.

Cliente – Servidor. El software se reparte en dos elementos diferentes pero sin reparto claro de funciones.

Arquitectura de tres niveles. Especialización de la arquitectura cliente – servidor, dividendo la carga en tres partes o capas, con un reparto claro de funciones. Capa para la presentación, capa para el cálculo y otra para el almacenamiento.

5.1.1 Arquitectura Cliente - Servidor

Esta arquitectura reparte el software en dos elementos, el cliente que realiza peticiones, y el servidor que le da respuesta. Esta idea se puede aplicar a programas que se ejecutan sobre una sola maquina, pero la forma más eficiente es separando las partes a través de una red de ordenadores, por lo que la capacidad de proceso esta repartida entre los clientes y los servidores.

La separación entre cliente servidor es una separación lógica, donde el servidor puede ejecutarse tanto en una maquina como en varias. Ejemplos de tipos de servidores pueden ser, servidores web, servidores de archivos, de correo…

Cliente (Parte activa)

- Demanda servicios a los servidores
- Se asume que cada petición deberá obtener respuesta
- Diseñado para interactuar con el usuario final
- Pueden conectarse varios a servidores a la vez

Servidor (Parte pasiva)

- Espera las peticiones de los clientes
- Procesa esas peticiones y envía las respuestas
- Diseño orientado a maximizar la eficiencia
- Normalmente no interactúan con el usuario final
- Aceptan conexiones desde un gran numero de clientes

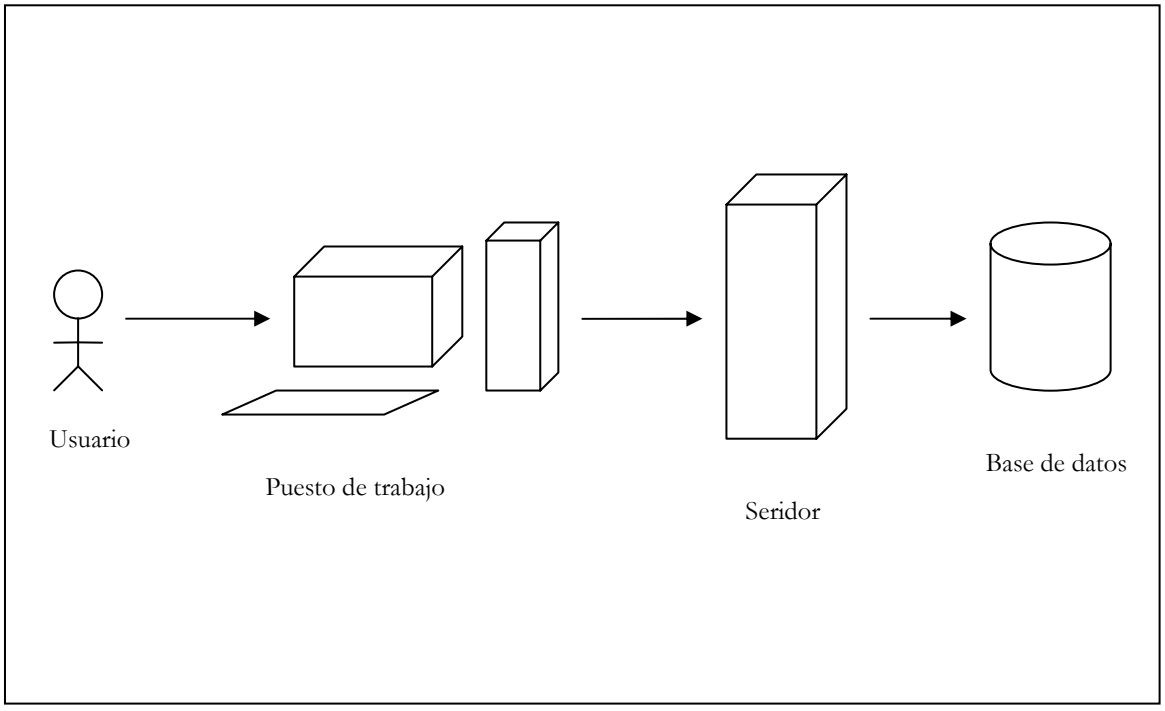

Ilustración 5-1 Arquitectura cliente – servidor

Puesto de trabajo o cliente

Una estación de trabajo o computador conectado a una red, que le permite acceder y gestionar una serie de recursos, perfilándose como un puesto de trabajo universal.

Favorece la flexibilidad y el dinamismo en las organizaciones.

Servidores

Una maquina que suministra una serie de servicios como Bases de Datos, Archivos, Comunicaciones…

Según su especialización pueden ser:

- Mainframes
- Miniordenadores
- Especializados

Una característica importante es que los diferentes servicios pueden ser suministrador por uno o varios servidores.

Existen tres capas dependiendo de su función dentro de la arquitectura.

- **Capa de presentación** (interfaz de usuario). Interacciona con el usuario, presenta los datos y recibe las entradas.
- **Capa de aplicación** (lógica de aplicación). Responsable de las tareas propias de la aplicación. Aplica las reglas del negocio sobre los datos y las entradas del usuario.
- **Capa de datos** (almacenamiento y acceso a datos). Responsable de la gestión y almacenamiento permanente de los datos.

Cada tipo de sistema cliente – servidor, distribuye estas capas de modo distinto entre los componentes existentes.

Ventajas

- **Centralización del control.** Todo se centra sobre el servidor, de forma que si algún programa cliente defectuoso o no autorizado no pueda dañar el sistema. Facilita también la actualización de los datos y otros recursos.
- **Escalabilidad.** Tanto clientes o servidores, pueden aumentar su capacidad independientemente del resto, incluso pudiendo añadir nuevos nodos a la red.
- **Fácil mantenimiento o encapsulación.** Al estar distribuidas las funciones entre varios ordenadores, es posible reparar, actualizar o migrar datos con mayor facilidad y sin que los clientes se vean afectados de una manera significativa.
- **Seguridad.** Existen tecnologías suficientes como para garantizar la seguridad en las transacciones.

Desventajas

- **Congestión del tráfico.** Si hay muchas peticiones a la vez por parte de los clientes, el servidor puede llegar a colapsarse si no las gestiona bien.
- **Robustez.** Si el servidor esta caído, las peticiones de los clientes no pueden ser satisfechas.
- **Software y Hardware.** Normalmente, se necesita un servidor potente para satisfacer el trabajo. Esta potencia, tiene un elevado coste respecto a software y hardware se refiere.
- **El cliente no dispone de los recursos que existen en el servidor.**

5.1.2 Tipos de clientes y servidores

Clientes ligeros

- No implementa ningún aspecto de la lógica de la aplicación
- Actúa como intermediario entre usuario y servidor.
  - recoge entradas y las envía al servidor
  - presenta datos y resultados del servidor
- Apenas necesita recursos hardware
- Aumenta la complejidad del servidor
- Clientes basados en navegadores Web (Jsp, Asp…) o capa de presentación repartida entre servidor (genera HTML al vuelo) y cliente (navegador)
- Aumento en los últimos años en el uso de clientes basados en navegador + soporte de interacciones complejas (java script, carga XML…)

Clientes pesados

- Implementa casi toda la lógica de la aplicación
- Realiza procesos sobres los datos antes de mandarlos al servidor
- Requiere equipos con capacidad de procesos y almacenamiento de datos
- Servidor sencillo

Clientes híbridos

- Implementación de la lógica repartida entre cliente y servidor

Servidor de archivos

- Servidor donde se almacenan archivos y aplicaciones de productividad

Servidor de bases de datos

- Servidor donde se almacenan bases de datos, tablas, índices, siendo uno de los servidores con mayor carga

Servidor de transacciones

- Servidor que cumple o procesa todas las transacciones. Primero valida y luego genera un pedido al servidor de bases de datos

Servidor de Groupware

- Servidor utilizado para el seguimiento de operaciones dentro de la red

Servidor de objetos

- Contiene objetos que deben estar fuera de la base de datos, imágenes, videos,…

Servidores Web

- Permite transacciones a través de un navegador. Es utilizado como comunicación entre empresas a través de Internet

5.2 Diagrama de Casos de Uso

El diagrama de casos de uso es una representación gráfica de parte o el total de los actores o casos de uso del sistema, representando la forma con el que los actores operan con el sistema, además de la forma, tipo y orden en como los elementos interactúan.

El modelado de Casos de Uso es una técnica sencilla y efectiva para el modelado de sistemas. Realmente no es una aproximación a la orientación a objetos, sino una forma de modelar los procesos. Forman parte del análisis del sistema y no del diseño como cabría pensar.

Elementos de un diagrama

- **Sistema**

  Un sistema se representa mediante una caja junto con el nombre del sistema arriba o dentro de la caja, definiendo así a aplicación que se quiere desarrollar., además de estableciendo unos limites entre lo que es interno y lo que es externo al sistema.

- **Caso de uso**

  Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema. Se representa por una elipse, y denota un requerimiento solucionado por el sistema. Cada caso de uso es una operación completa desarrollada por los actores o por el sistema. El nombre de caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

- **Actor**

  Un actor es una entidad que utiliza alguno de los casos de uso del sistema. Se representa mediante un muñeco acompañado de un nombre significativo. El actor interactúa con el sistema enviando o recibiendo del sistema unos mensajes o intercambiando información con este.

  Un actor es algo con comportamiento, lo que se podría definir como un rol, que especifica que un actor no debe ser necesariamente una persona.

Relaciones entre Casos de Uso

Un Caso de uso, en principio, debería describir una tarea que tiene un sentido completo para el usuario. Sin embargo, hay ocasiones en las que es útil describir una interacción con un alcance menor como caso de uso. La razón para utilizar estos casos de uso no completos en algunos casos, es mejorar la comunicación en el equipo de desarrollo, el manejo de la documentación de casos de uso.

- Include

  Un caso de uso base incorpora explícitamente a otro caso de uso en un lugar especificado en dicho caso base. Es una simple relación de inclusión, es decir, los

escenarios o situaciones posibles detalladas en un caso de uso están incluidas en otro caso de uso.

- Extend

  Cuando un caso de uso base tiene ciertos puntos en los cuales, dependiendo de ciertos criterios, se va a realizas una interacción adicional. El caso de uso que extiende describe un comportamiento opcional del sistema. Las secuencias alternas se modelan en casos de uso separados, los cuales están relaciones con el caso de uso original mediante una relación "Extiende". Además se suele utilizar para representar casos de uso que engloban a otros.

- Generalización

  Cuando un caso de uso definido de forma abstracta se particulariza por medio de otro caso de uso más especifico. Al igual que en la herencia entre clases, el caso de uso hijo hereda las asociaciones y características del caso de uso padre. También se puede dar entre actores.

5.2.1 Diagrama de Casos de Uso del sistema

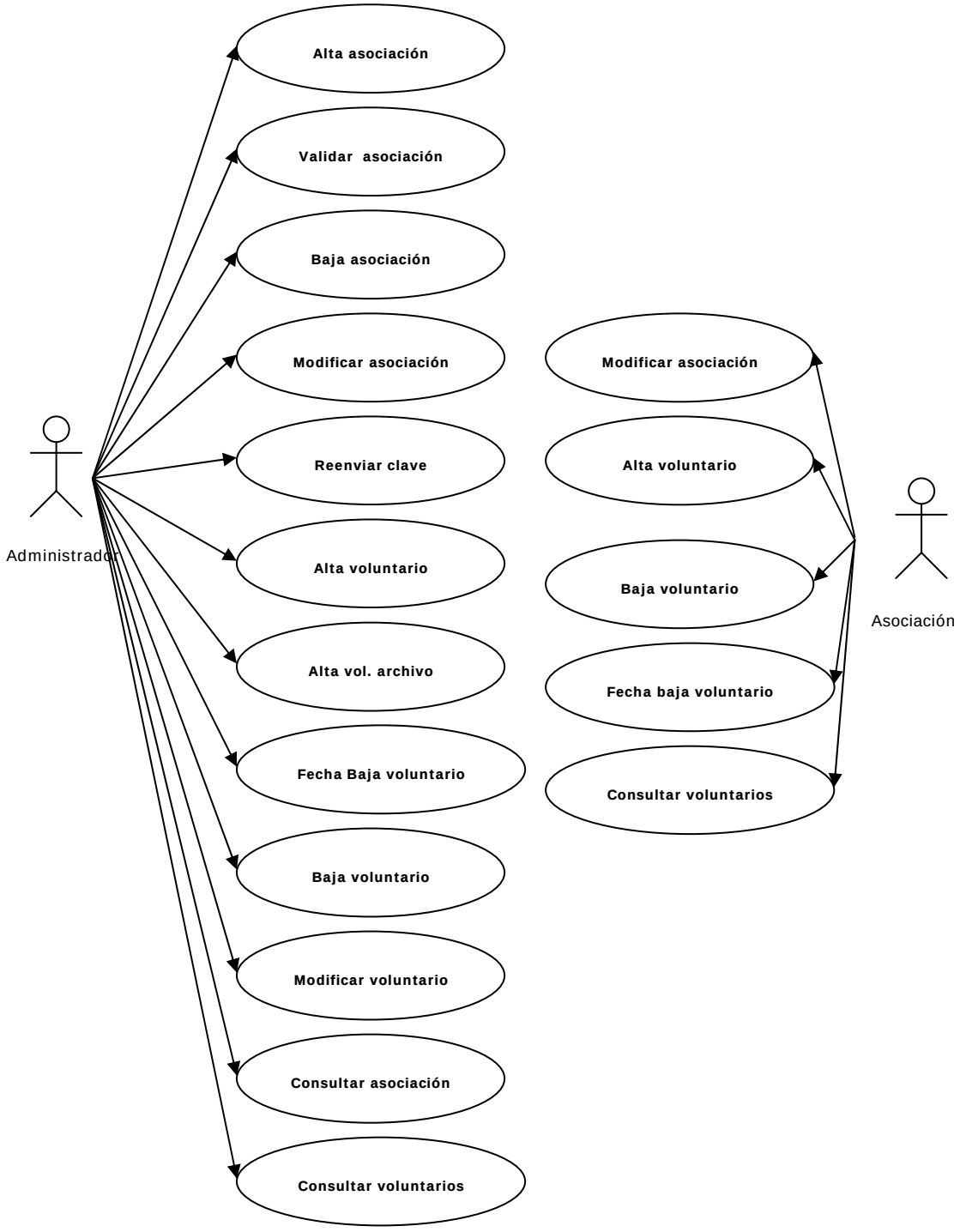

Ilustración 5-2 Diagrama casos de uso

5.3 Explicaciones de los Casos de Uso

5.3.1 Casos de Uso Administrador

Nombre:	**Alta asociación**
Descripción: Dar de alta a una asociación en el sistema.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: Asociación dada de alta en el sistema.	
Flujo Normal: • El administrador pulsa la opción "Nueva Asociación". • El administrador rellena todos los campos necesarios para el alta de la asociación. • El sistema muestra de nuevo los datos por si hubiese algún fallo. • El administrador acepta los datos. • El sistema verifica que los campos estén correctamente introducidos. • El sistema informa de la nueva alta. • El sistema envía los correos correspondientes a la alta de asociación.	
Flujo Alternativo: • El sistema detecta incorrecciones en los campos introducidos. • El sistema detecta que existe otra asociación con el mismo CIF.	

Tabla 17. Caso de Uso Alta asociación

Nombre:	**Validar asociación**
Descripción: Validar la asociación en el sistema para que pueda gestionar ella misma los voluntarios.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema. Asociación dada de alta en el sistema.	
Poscondiciones: Asociación validada en el sistema.	
Flujo Normal: • El administrador pulsa la opción "Validar Asociación". • El administrador rellena todos los campos necesarios para la validación. • El administrador acepta los datos. • El sistema verifica que los campos estén correctamente introducidos. • El sistema informa de la correcta validación. • El sistema envía el nombre de usuario y la clave a la asociación.	
Flujo Alternativo: • El sistema detecta incorrecciones en los campos introducidos.	

Tabla 18. Caso de Uso Validar asociación

Nombre:	**Baja asociación**
Descripción: Dar de baja a una asociación en el sistema.	
Actores: Administrador.	
Precondiciones:	

El administrador debe de estar dentro del sistema.
Poscondiciones: Asociación dada de baja en el sistema.
Flujo Normal: • El administrador pulsa la opción "Baja Asociación". • El sistema muestra el listado de asociaciones disponibles en la base de datos. • El administrador selecciona aquellas que desea dar de baja. • El administrador acepta la selección. • El sistema realiza la baja de las asociaciones.
Flujo Alternativo: • El administrador introduce el CIF de la empresa a dar de baja. • EL sistema comprueba la existencia del CIF. • Si existe, la da de baja, si no, informa de CIF inexistente.

Tabla 19. Caso de Uso Baja asociación

Nombre:	**Modificar asociación**
Descripción: Modificar los datos de la asociación.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: Asociación modificada correctamente.	
Flujo Normal: • El administrador pulsa la opción "Modificar Asociación". • El sistema muestra el listado de asociaciones disponibles en la base de datos. • El administrador selecciona aquella que desea modificar. • El administrador acepta la selección. • El sistema muestra los datos de la asociación. • El administrador modifica aquellos que considere necesarios.	

• El administrador valida los cambios. • El sistema comprueba que los cambios sean correctos. • El sistema modifica los datos. • El sistema envía los correos pertinentes con los cambios realizados.
Flujo Alternativo: • El sistema detecta que hay datos incorrectos.

Tabla 20. Caso de Uso Modificar asociación

Nombre:	**Reenviar clave asociación**
Descripción: El administrador reenvía la clave de acceso a la asociación.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: Clave reenviada.	
Flujo Normal: • El administrador pulsa la opción "Modificar Asociación". • El sistema muestra el listado de asociaciones disponibles en la base de datos. • El administrador selecciona aquella que desea modificar. • El administrador acepta la selección. • El sistema muestra los datos de la asociación. • El administrador pulsa la opción "Reenviar clave". • El sistema reenvía la clave al correo de la asociación.	
Flujo Alternativo:	

Tabla 21. Caso de Uso Reenviar clave asociación

Nombre:	**Alta voluntario**
Descripción: Dar de alta a uno o varios voluntarios en el sistema.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: Voluntarios añadidos.	
Flujo Normal: • El administrador pulsa la opción "Alta Voluntario". • El sistema muestra las tablas donde dar de alta a los voluntarios y la lista de asociaciones. • El administrador rellena los datos de los voluntarios. • El administrador selecciona la asociación en la que desea añadirlos. • El sistema comprueba que los datos estén introducidos correctamente. • El administrador valida los datos. • El sistema realiza el alta de los voluntarios en la base de datos. • El sistema informa tanto por correo como por pantalla de los voluntarios añadidos junto con sus certificados correspondientes.	
Flujo Alternativo: • El sistema detecta fallos en los datos de los voluntarios. • El sistema detecta voluntarios con el mismo NIF. • El sistema informa que hay voluntarios que no se han dado de alta por tener el mismo NIF.	

Tabla 22. Caso de Uso Alta voluntario

Nombre:	**Alta Voluntario desde archivo**
Descripción: Alta de voluntarios a través de archivo externo.	
Actores:	

Administrador.
Precondiciones: El administrador debe de estar dentro del sistema.
Poscondiciones: Alta de voluntarios..
Flujo Normal: • El administrador pulsa la opción "Alta desde archivo". • El sistema muestra el listado de asociaciones disponibles en la base de datos junto con el campo donde deberá introducir la ubicación del archivo. • El administrador selecciona el archivo y la asociación en la que realizar el alta. • El administrador acepta la selección. • El sistema comprueba que los voluntarios tengan el formato correcto. • El sistema da de alta a los voluntarios en la base de datos. • El sistema envía un correo con los voluntarios dados de alta.
Flujo Alternativo: • El sistema detecta incorrecciones en los datos de los voluntarios. • El sistema informa detalladamente de los datos incorrectos por pantalla. • El administrador corrige los datos en el archivo y vuelve al flujo normal.

Tabla 23. Caso de Uso Alta Voluntarios desde archivo

Nombre:	**Baja Voluntario**
Descripción: Da de baja a los voluntarios del sistema.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: Baja de voluntarios.	

Flujo Normal: • El administrador pulsa la opción "Baja voluntarios". • El sistema muestra las asociaciones disponibles en la base de datos. • El administrador selecciona una de ellas. • El sistema muestra los voluntarios pertenecientes a dicha asociación. • El administrador selecciona aquellos voluntarios que desea dar de baja. • El administrador confirma la baja de los voluntarios. • El sistema da de baja a los voluntarios.
Flujo Alternativo: • El sistema detecta que el administrador no ha seleccionado ningún voluntario.

Tabla 24. Caso de Uso Baja voluntario

Nombre:	**Fecha Baja Voluntario**
Descripción: Establece le fecha de baja de los voluntarios.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: Fecha de baja modificadas.	
Flujo Normal: • El administrador pulsa la opción "Baja voluntarios". • El sistema muestra las asociaciones disponibles en la base de datos. • El administrador selecciona una de ellas. • El sistema muestra los voluntarios pertenecientes a dicha asociación. • El administrador selecciona aquellos voluntarios en los que desea establecer una fecha de baja. • El administrador introduce una fecha de baja. • El administrador confirma la selección. • El sistema establece la fecha de baja correspondiente para los voluntarios elegidos. • El sistema envía un correo con los voluntarios modificados y la fecha de baja establecida.	
Flujo Alternativo:	

• El sistema detecta que el administrador no ha seleccionado ningún voluntario. • El sistema detecta que la fecha introducida no tiene el formato correcto.

Tabla 25. Caso de Uso Fecha baja voluntario

Nombre:	**Modificar Voluntario**
Descripción: Modifica la información de los voluntarios de la base de datos.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: Datos del voluntario modificados.	
Flujo Normal: • El administrador pulsa la opción "Modificar Voluntario". • El sistema muestra las asociaciones disponibles en la base de datos. • El administrador selecciona una de ellas. • El sistema muestra los voluntarios pertenecientes a dicha asociación. • El administrador selecciona un voluntario para modificar. • El sistema muestra los datos correspondientes a dicho voluntario. • El administrador modifica los datos necesarios. • El sistema comprueba que los datos establecidos son correctos. • El sistema actualiza los datos del voluntario en la base de datos.	
Flujo Alternativo: • El sistema detecta que los datos introducidos son incorrectos.	

Tabla 26. Caso de Uso Modificar voluntario

Nombre:	**Consultar Asociaciones**
Descripción: Realiza una consulta de las asociaciones sobre la base de datos.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones: -	
Flujo Normal: • El administrador pulsa la opción "Modificar Voluntario". • El sistema muestra un formulario con los campos de las asociaciones. • El administrador rellena aquellos que considere necesarios para realizar la búsqueda. • El sistema realiza la consulta y muestra por pantalla la información correspondiente a las asociaciones que cumplan los criterios de búsqueda.	
Flujo Alternativo: -	

Tabla 27. Caso de Uso Consultar asociaciones

Nombre:	**Consultar Voluntarios**
Descripción: Realiza una consulta de los voluntarios sobre la base de datos.	
Actores: Administrador.	
Precondiciones: El administrador debe de estar dentro del sistema.	
Poscondiciones:	

-
Flujo Normal: • El administrador pulsa la opción "Modificar Voluntario". • El sistema muestra un formulario con los campos de los voluntarios. • El administrador rellena aquellos que considere necesarios para realizar la búsqueda. • El sistema realiza la consulta y muestra por pantalla la información correspondiente a los voluntarios que cumplan los criterios de búsqueda.
Flujo Alternativo: -

Tabla 28. Caso de Uso Consultar voluntarios

5.3.2 Casos de uso Asociación

Nombre:	**Modificar asociación**
Descripción: Modificar los datos de la asociación.	
Actores: Asociación.	
Precondiciones: La asociación debe de estar dentro del sistema.	
Poscondiciones: Asociación modificada correctamente.	
Flujo Normal: • La asociación pulsa la opción "Modificar Asociación". • El sistema muestra los datos de la asociación. • La asociación modifica aquellos que considere necesarios. • La asociación valida los cambios. • El sistema comprueba que los cambios sean correctos. • El sistema modifica los datos.	

• El sistema envía los correos pertinentes con los cambios realizados.
Flujo Alternativo:
• El sistema detecta que hay datos incorrectos.

Tabla 29. Caso de Uso Modificar asociación

Nombre:	**Alta voluntario**
Descripción: Dar de alta a uno o varios voluntarios en el sistema.	
Actores: Asociación.	
Precondiciones: La asociación debe de estar dentro del sistema.	
Poscondiciones: Voluntarios añadidos.	
Flujo Normal: • La asociación pulsa la opción "Alta Voluntario". • El sistema muestra las tablas donde dar de alta a los voluntarios. • La asociación rellena los datos de los voluntarios. • El sistema comprueba que los datos estén introducidos correctamente. • La asociación valida los datos. • El sistema realiza el alta de los voluntarios en la base de datos. • El sistema informa tanto por correo como por pantalla de los voluntarios añadidos junto con sus certificados correspondientes.	
Flujo Alternativo: • El sistema detecta fallos en los datos de los voluntarios. • El sistema detecta voluntarios con el mismo NIF. • El sistema informa que hay voluntarios que no se han dado de alta por tener el mismo NIF.	

Tabla 30. Caso de Uso Alta voluntario

Nombre:	**Baja Voluntario**
Descripción: Da de baja a los voluntarios del sistema.	
Actores: Asociación.	
Precondiciones: La asociación debe de estar dentro del sistema.	
Poscondiciones: Baja de voluntarios.	
Flujo Normal: • La asociación pulsa la opción "Baja voluntarios". • El sistema muestra los voluntarios pertenecientes a la asociación. • La asociación selecciona aquellos voluntarios que desea dar de baja. • La asociación confirma la baja de los voluntarios. • El sistema da de baja a los voluntarios.	
Flujo Alternativo: • El sistema detecta que la asociación no ha seleccionado ningún voluntario.	

Tabla 31. Caso de Uso Baja voluntario

Nombre:	**Fecha Baja Voluntario**
Descripción: Establece le fecha de baja de los voluntarios.	
Actores: Asociación.	
Precondiciones: La asociación debe de estar dentro del sistema.	

Poscondiciones: Fecha de baja modificadas.
Flujo Normal: • La asociación pulsa la opción "Baja voluntarios". • El sistema muestra los voluntarios pertenecientes a la asociación. • La asociación selecciona una de ellas. • La asociación selecciona aquellos voluntarios en los que desea establecer una fecha de baja. • La asociación introduce una fecha de baja. • La asociación confirma la selección. • El sistema establece la fecha de baja correspondiente para los voluntarios elegidos. • El sistema envía un correo con los voluntarios modificados y la fecha de baja establecida.
Flujo Alternativo: • El sistema detecta que la asociación no ha seleccionado ningún voluntario. • El sistema detecta que la fecha introducida no tiene el formato correcto.

Tabla 32. Caso de Uso Fecha baja voluntario

Nombre:	**Consultar Voluntarios**
Descripción: Realiza una consulta de los voluntarios sobre la base de datos.	
Actores: Asociación.	
Precondiciones: La asociación debe de estar dentro del sistema.	
Poscondiciones: -	
Flujo Normal: • La asociación pulsa la opción "Modificar Voluntario". • El sistema muestra un formulario con los campos de los voluntarios. • La asociación rellena aquellos que considere necesarios para realizar la búsqueda. • El sistema realiza la consulta y muestra por pantalla la información correspondiente a los voluntarios que cumplan los criterios de búsqueda.	

Flujo Alternativo: -

Tabla 33. Caso de Uso Consultar voluntarios

5.4 Modelo Entidad Relación de la base de datos

El modelo entidad relación es un modelo conceptual para la creación de base de datos, propuesto en 1976 por Peter Chen. El objetivo de es modelo es poder visualizar los objetos que pertenecen a la base de datos como entidades, junto con sus atributos y relaciones.

En el modelo, se pueden representar:

- los datos vistos como entidades
- atributos o características de dichas entidades
- relaciones entre ellas
- cierta semántica del problema
- ciertas restricciones

5.4.1 Elementos

- **Entidad**

  Es un objeto del mundo real que es distinguible de todos los demás. Puede tratarse de cualquier tipo de objeto, persona, empresa, cosa, informe, etc.

  Las entidades se representan gráficamente como rectángulos, apareciendo su nombre en el interior. Las entidades a su vez, tienen atributos o propiedades.

  Existen dos tipos de entidades, fuertes y débiles. Una entidad fuerte es independiente de la existencia de otra, mientras que una entidad débil depende de la existencia de otra.

- **Atributos**

  Los atributos describen propiedades que poseen cada entidad o relación. Gráficamente se representan mediante unos círculos unidos mediante líneas a los demás objetos.

  Cada atributo posee un conjunto de valores asociados llamado dominio. El dominio se define como los valores posibles que puede tomar dicho atributo.

Se pueden distinguir varios tipos de atributos:

Simples o compuestos. Un atributo simple es aquel que tiene un solo componente, es decir, es indivisible en partes más pequeñas. Un atributo compuesto es un atributo que se puede dividir en otros pequeños con significado propio, y que mantienen una afinidad común.

Derivados. Un atributo derivado es aquel que representa un valor formado a partir del valor de uno o varios atributos.

Monovalorados o multivalorados. Un atributo monovalorado es aquel que solo tiene un valor para cada ocurrencia de la entidad o relación, como por ejemplo la altura en una persona. Un atributo multivalorado tiene por el contrario varios valores para cada ocurrencia de la relación o entidad, como puede ser el número de hijos.

- **Relación**

  Es una correspondencia o asociación entre dos o más entidades. Cada relación describe su función mediante un nombre. Su representación gráfica es un rombo con el nombre en el interior.

  Las entidades que están involucradas en una determinada relación se denominan entidades participantes. El número de participantes en una relación es lo que se denomina grado de la relación. Por lo tanto, una relación en l que participan dos entidades es una relación binaria. Si son tres las entidades será una relación ternaria, etc.

  La cardinalidad es con la que una entidad participa en una relación específica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menor, una ocurrencia de la otra entidad participante. Si no, la participación es opcional (parcial).

- Identificador

  Es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad. UN identificador de una entidad tiene que cumplir:

  1.- No pueden existir dos ocurrencias de la entidad con el mismo identificador.

  2.- Si se omite cualquier atributo del identificador, la condición anterior deja de cumplirse.

  Toda entidad tiene al menos un identificador y puede tener varios identificadores alternativos. Las relaciones no tienen identificadores.

5.4.2 Modelo Entidad Relación de la base de datos

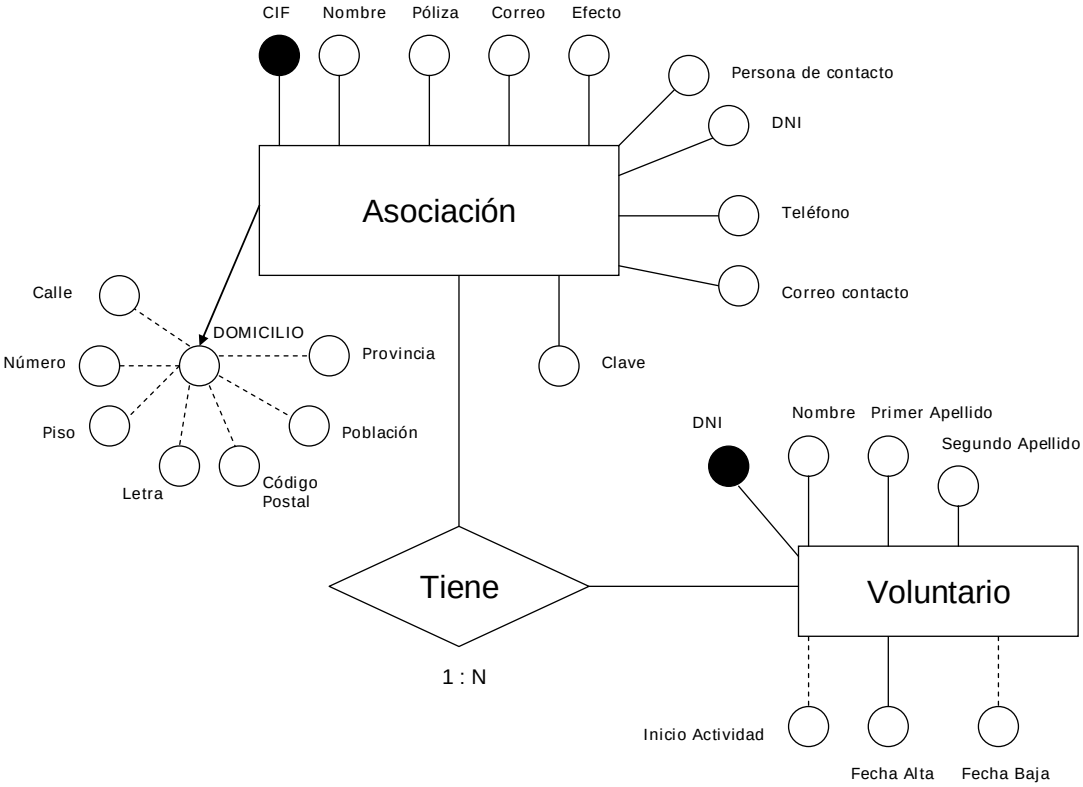

Ilustración 5-3 Modelo entidad relación BBDD

5.5 Modelo Relacional de la base de datos

Este modelo ofrece una visión más textual y esquemática de cómo será la base de datos. Esta basado en la lógica de predicados y en la teoría de conjuntos. Actualmente es el modelo más usado para resolver problemas reales y administrar datos dinámicamente.

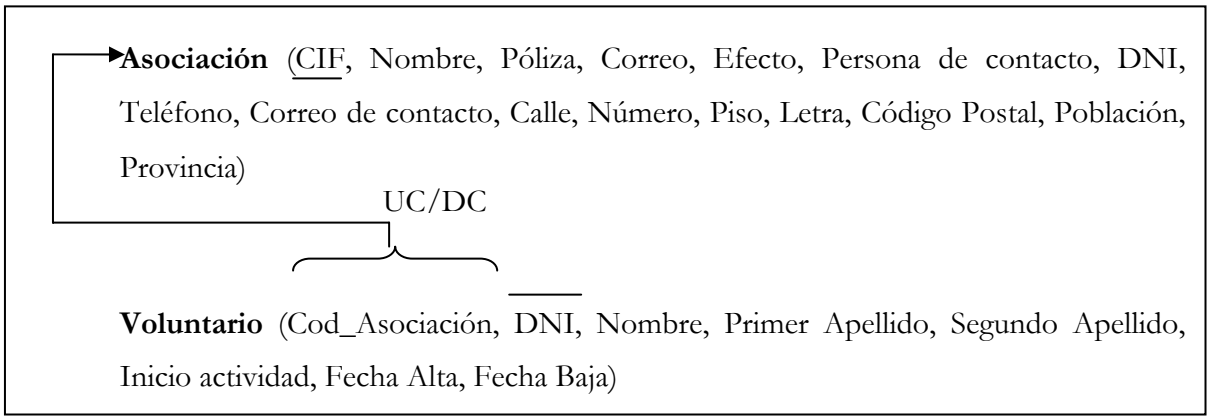

Ilustración 5-4 Modelo relacional BBDD

5.5.1 Explicación de Entidades y Atributos de la BBDD

En este apartado se explican las diferentes tablas de las que se compone la base de datos, así como los atributos de cada una de ellas.

Tabla Asociación

Atributos	Tipo de datos	Nulo	Predeterminado	Descripción
Nombre	VARCHAR (80)	NO		Nombre de la asociación
Póliza	VARCHAR (80)	NO		Póliza de la asociación
Efecto	DATE	NO		Fecha de efecto de la póliza
CIF (PK)	VARCHAR (10)	NO		CIF de la asociación
Calle	VARCHAR (80)	SI	NULL	Calle de residencia
Número	VARCHAR (3)	SI	NULL	Número de la calle
Piso	VARCHAR (3)	SI	NULL	Número de piso
Letra	VARCHAR (2)	SI	NULL	Letra de piso
Código Postal	VARCHAR (8)	SI	NULL	Código postal de residencia
Población	VARCHAR (60)	SI	NULL	Población de la asociación
Provincia	VARCHAR (60)	SI	NULL	Provincia de la asociación
Correo Asociación	VARCHAR (120)	NO		Correo de la asociación
Persona	VARCHAR (120)	SI	NULL	Persona de contacto
NIF	VARCHAR (11)	SI	NULL	NIF de la persona
Teléfono	VARCHAR (15)	SI	NULL	Teléfono de la persona
Correo	VARCHAR (120)	SI	NULL	Correo de la persona
Clave	VARCHAR (20)	NO		Clave de acceso

Tabla 34 Descripción detallada de los atributos de Asociación

Tabla Voluntario

Atributos	Tipo de datos	Nulo	Predeterminado	Descripción
Nombre	VARCHAR (60)	NO		Nombre del voluntario
Apellido 1	VARCHAR (60)	NO		Primer apellido del voluntario
Apellido 2	VARCHAR (60)	SI	NULL	Segundo apellido del voluntario
DNI (PK)	VARCHAR (15)	NO		DNI del voluntario
Fecha Inicio	DATE	SI	NULL	Fecha de Inicio del seguro
Fecha alta	DATE	SI	NULL	Fecha de alta en la aplicación
Fecha baja	DATE	SI	NULL	Fecha de baja del seguro
Asociación (FK)	VARCHAR (10)	NO		CIF de la asociación

Tabla 35 Descripción detallada de los atributos de Voluntario

5.6 Diagrama de clases

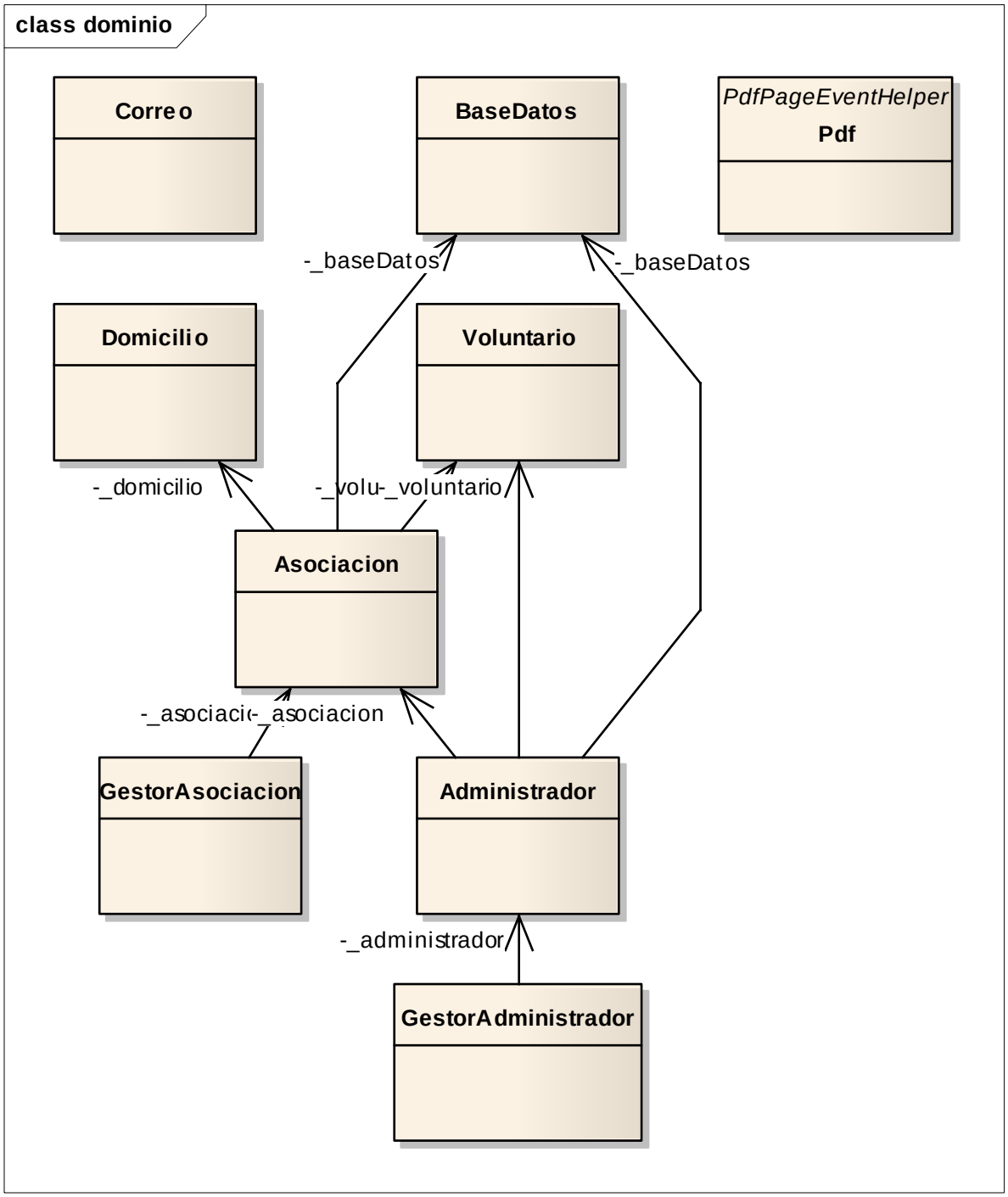

Ilustración 5-5 Diagrama de clases 1

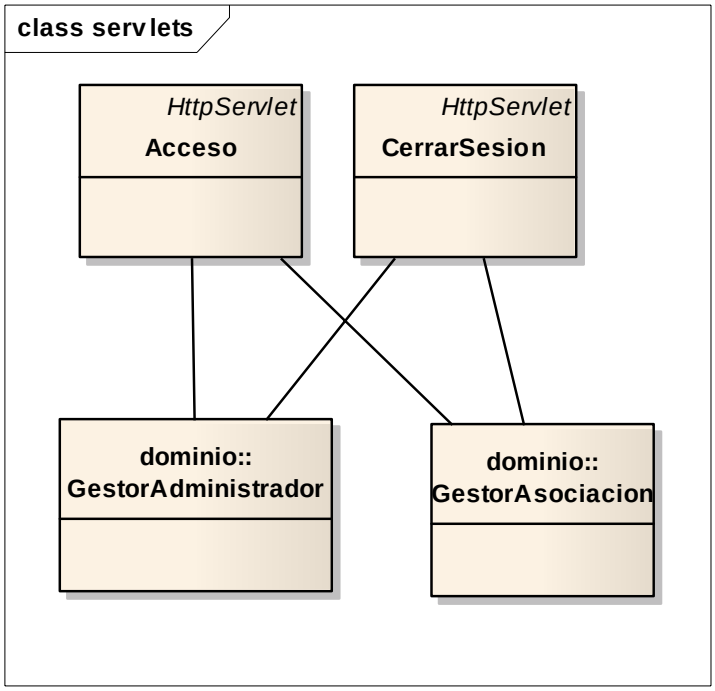

Ilustración 5-6 Diagrama de clases 2

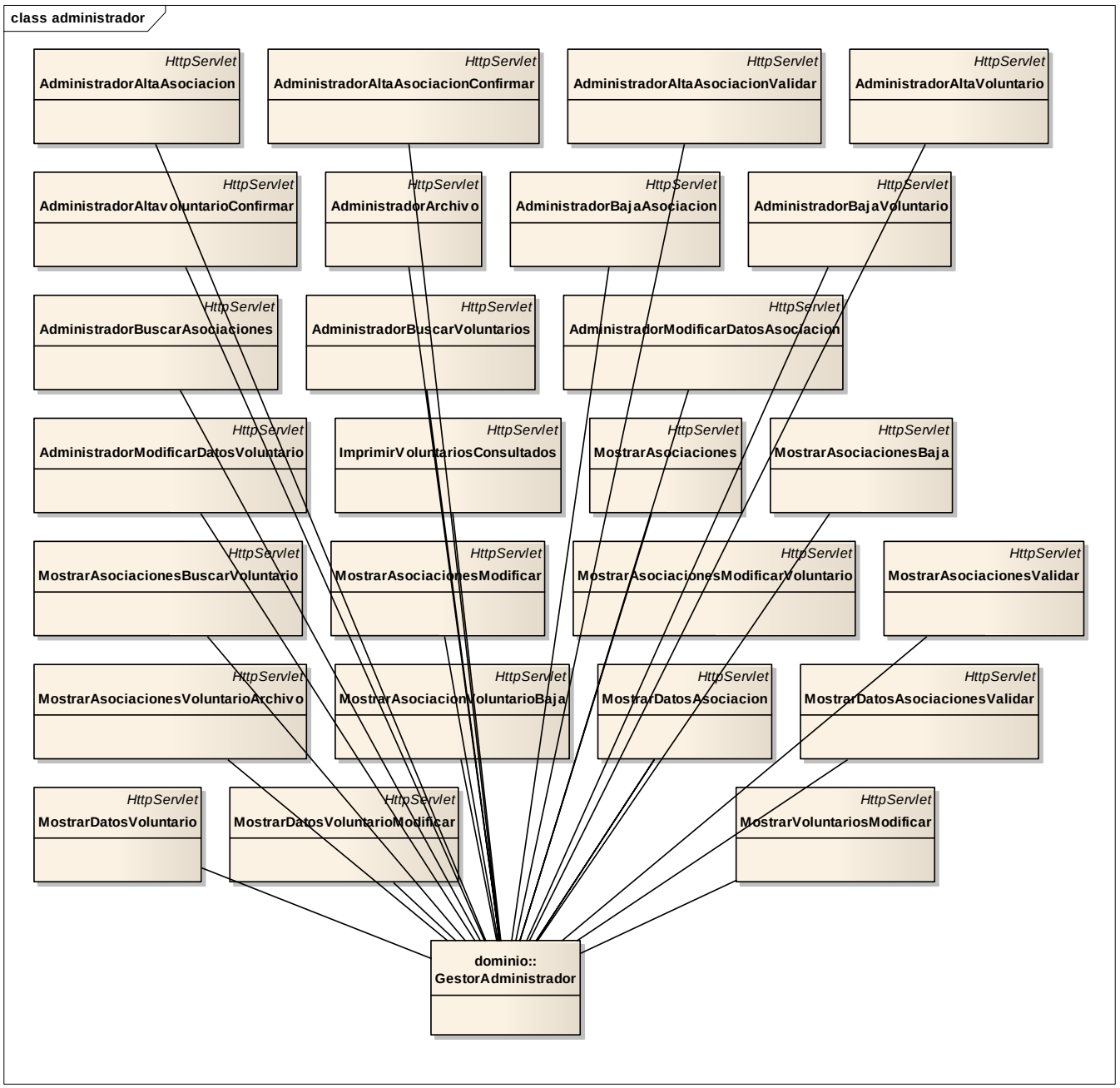

Ilustración 5-7 Diagrama de clases 3

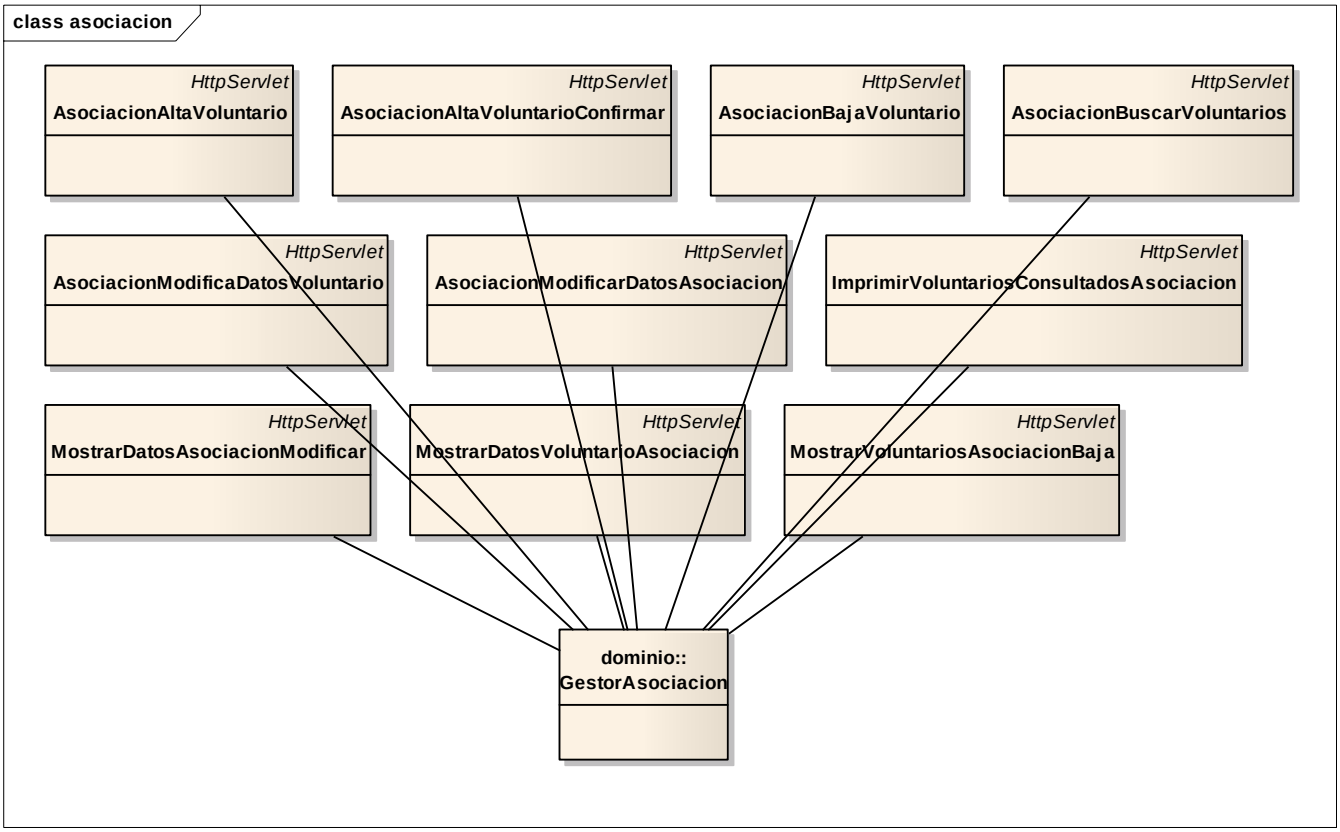

Ilustración 5-8 Diagrama de clases 4

6 Desarrollo

6.1 Introducción

La fase de desarrollo corresponde con la implementación y documentación de la aplicación. Para ello, en primer lugar se ha requerido la elección de una plataforma de desarrollo, en este caso, NetBeans, que permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software (módulos).

Una vez elegida la plataforma, el lenguaje de programación estaba claro, Java. Se trata de un lenguaje ya conocido y muy utilizado en las aplicaciones Web. Además, permite añadir nuevas librerías gratuitas, tales como IText o Javamail, utilizadas en el desarrollo de la aplicación, y fundamentales para cubrir las necesidades requeridas. Más adelante se verán algunos ejemplos del código utilizado.

Respecto a la documentación, se ha ido realizando al mismo tiempo que la codificación, evitando así posteriores revisiones al código para ver cual era el funcionamiento de un método. Se ha generado Javadoc sobre todos los métodos programados para la aplicación, creando así una pequeña API en donde se explican brevemente el funcionamiento de cada uno de ellos.

El tiempo de desarrollo ha sido algo superior al estimado en un principio. El principal motivo fue el aprendizaje paralelo que se iba haciendo de las nuevas librerías a utilizar, lo que produjo constantes pruebas y modificaciones en el código hasta dar con el resultado esperado.

Finalmente, el desarrollo del proyecto se llevo a cabo satisfactoriamente, cumpliendo todos los requisitos establecidos, e intentando generar un código y una documentación lo más reutilizable y eficiente posible, con vistas a recuperar este desarrollo y utilizarlo como base a una más que probable futura aplicación.

6.2 Explicación código IText

A continuación se exponen ciertas partes del código utilizado para generar PDF a través de los datos obtenidos de la Base de Datos.

Crear un documento:

```
public void abrirDocumento() {
        this.doc = new Document();
        try {
            PdfWriter.getInstance(this.doc, this.response.getOutputStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
         this.doc.open();
    }
```

Ilustración 6-1 Código para crear un documento

En primer lugar se crea una instancia de la clase Document, que será donde se irá añadiendo toda la información del documento.

Una vez creado el documento, se crea una instancia de PdfWriter, que será el encargado de escribir la información sobre el flujo de salida escogido. Si no se genera ninguna excepción, se procederá a abrir el documento.

Establecer que el documento sea un PDF.:

```
this.response.setContentType("application/pdf");
```

Ilustración 6-2 Código para establecer formato de documento

Para crear un archivo PDF, debemos establecer que la salida, en este caso el documento, sea en formato PDF. Para ello se utiliza el objeto response, perteneciente a los Servlets, y encargado de la respuesta de la aplicación.

Crear una tabla y rellenarla:

```
PdfPTable t = new PdfPTable(2);
PdfPCell c;= new PdfPCell(new Paragraph("\n" + "            VOLUNTARIOS
" + "\n" + "\n", time));
c.setBackgroundColor(Color.lightGray);
c.setHorizontalAlignment(c.ALIGN_CENTER);
t.addCell(c);
```

Ilustración 6-3 Código para crear una tabla

Para crear una tabla, primero se debe instanciar un objeto PdfTable, cuyo parámetro de construcción será el número de columnas que se desean tener.

Para rellenarla, se deben ir creando celdas, en este caso, PdfCell, añadiendo el contenido de la celda y el formato de fuente. Más tarde, se definirá el color de fondo y su alineación.

Insertar una imagen:

```
URL au = new URL("http://www.flexibletree.com/images/mapfre.JPG");
Image ima = Image.getInstance(au);
ima.setAlignment(ima.ALIGN_LEFT);
ima.scalePercent(80, 80);
ima.setAbsolutePosition(10, 750);
document.add(ima);
```

Ilustración 6-4 Código insertar una imagen

Para insertar una imagen, en primer lugar se debe indicar la dirección donde se encuentra, en este caso, una dirección web. Posteriormente, se instancia un objeto imagen con la dirección anterior. Los siguientes métodos, definen la situación y el tamaño de la imagen dentro del documento. Por último, solo queda añadirla al documento.

Insertar un párrafo:

```
Paragraph p = new Paragraph("Parrafo", cou2);
p.setAlignment(p.ALIGN_CENTER);
document.add(p);
```

Ilustración 6-5 Código insertar un párrafo

Si se desea insertar un párrafo en el documento, se deberá crear en primer lugar una instancia Párrafo, con el texto que se desea añadir y el tipo de letra en su constructor.

Posteriormente, se podrán definir las características que se quieran del párrafo, en este caso, su alineación. Y finalmente, añadirlo en el documento.

Crear un pie de página:

```
Rectangle page = document.getPageSize();
Font time2 = new Font(Font.HELVETICA, 6, Font.NORMAL);
String parrafo1 = null;
parrafo1 = " Correduría de Seguros.";
PdfPTable foot = new PdfPTable(1);
Paragraph p = new Paragraph(parrafo1, time2);
p.setAlignment(p.ALIGN_CENTER);
PdfPCell cell = new PdfPCell(p);
cell.setBorderColor(Color.WHITE);
cell.setHorizontalAlignment(cell.ALIGN_CENTER);
cell.setVerticalAlignment(cell.ALIGN_CENTER);
foot.addCell(cell);
foot.setHorizontalAlignment(foot.ALIGN_CENTER);
foot.setTotalWidth(page.getWidth() - document.leftMargin() - document.rightMargin());
writer.getDirectContent());
docWriter.setPageEvent(new Pdf());
```

Ilustración 6-6 Código insertar un pie de página

Esta tarea tal vez sea la que agrupa más elementos a utilizar. Como se puede ver, hay un párrafo, que será el que contenga el texto definido. Una celda donde meter el párrafo y poder recuadrarlo, perteneciente a la tabla PdfTable, la que definirá en qué posición debe ir el pie de página.

La última línea, será la que creará una nueva página y un nuevo pie de página cuando reciba un evento de docWriter.

6.3 Explicación código JavaMail

A continuación se explican algunos fragmentos de código utilizados para la creación y envío de correos a través de la librería JavaMail.

Constructor de Correo:

```
public Correo(String host, String from) {
        this.props = new Properties();
        this.props.setProperty("mail.smtp.host", host);
        this.props.setProperty("mail.smtp.starttls.enable", "true");
        this.props.setProperty("mail.smtp.port", "25");
        this.props.setProperty("mail.smtp.user", from);
        this.props.setProperty("mail.smtp.auth", "true");
```

Ilustración 6-7 Código constructor correo

En primer lugar, se debe crear un nuevo correo a través de esto constructor, donde se definirán las propiedades requeridas para que el envío sea satisfactorio. Se deben establecer cinco propiedades, la primera de ellas será el host del servidor de correos, que serán pasados como argumento junto con el remitente del correo, es decir, la dirección de correo desde la que se mandará. El resto de atributos serán específicos para cada servidor, en este caso, se utilizará el puerto 25, el servicio de autentificación debe estar activo igual que starttls.

Enviar correo:

```
public void enviar(String to, String de, String asunto, String clave, String texto) throws Exception
{
        try {
                Session session = Session.getDefaultInstance(this.props);
                MimeMessage message = new MimeMessage(session);
                message.setFrom(new InternetAddress(de));
                message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
                message.setSubject(asunto);
                message.setText(this._texto, "ISO-8859-1", "html");
                Transport t = session.getTransport("smtp");
       try {
                t.connect(this.props.getProperty("host"), de, clave);
                t.sendMessage(message, message.getAllRecipients());
       } catch (Exception e) {
                throw new Exception("No se ha podido mandar el correo, correo incorrecto");}
        t.close();
```

Ilustración 6-8 Código enviar correo

Para poder enviar correos se debe crear en primer lugar una Session con las propiedades anteriormente definidas en el constructor.

Posteriormente se crea el mensaje, con el paso de sesión en el constructor. Se definen en este objeto el remitente, destinatario, asunto, texto y formato de texto.

Finalmente, se crea un objeto transport con el nombre del protocolo a utilizar. Se establece la conexión mediante el nombre de usuario y la contraseña, y se envía el mensaje a los destinatarios seleccionados. Una vez acabado, se cierra la conexión.

Enviar correo con archivo adjunto:

```
public Multipart incluirFichero(Multipart multipart, BodyPart messageBodyPart, Vector ficheros) throws Exception {
        int i = 0;
        ByteArrayOutputStream bs = null;
        while (i < ficheros.size()) {
                try {
                messageBodyPart = new MimeBodyPart();
                bs = (ByteArrayOutputStream) ficheros.elementAt(i);
                messageBodyPart.setDataHandler(newDataHandler(new
        ByteArrayDataSource(bs.toByteArray(), "application/pdf")));
                messageBodyPart.setFileName("documento" + i + ".pdf");
                i++;
                multipart.addBodyPart(messageBodyPart);
                bs.reset();
                } catch (Exception e) {
                 throw new Exception("Error al incluir el fichero");
                }
         }
        bs.close();
```

Ilustración 6-9 Código enviar correo con adjunto

Si se desea enviar archivos a través de un correo, se debe crear un objeto que una el cuerpo del mensaje (MimeBodyPart) con el archivo que se quiere adjuntar (DataHandler. ByteArrayDataSource), especificando en este ultimo el tipo de documento que se quiere obtener, en este caso, un archivo PDF.

Una vez instanciadas estas partes, se creara un ByteArrayOutputStream, el buffer de salida que contiene toda la información del archivo.

El proceso de creación de archivo se iterará tantas veces como archivos se quieran adjuntar, y en todas ellas los pasos a seguir serán los siguientes:

- Creación del cuerpo del mensaje
- Se pasa al buffer de salida el contenido el fichero, obtenido a partir de un Vector.
- Se crea la clase que maneje los datos, DataHandler, pasandole como argumentos la información y el tipo de fichero.
- Se etiqueta el archivo mediante un nombre.
- Añade el cuerpo del mensaje al mensaje multiparte.
- Iteración del bucle.
- Reset al buffer para prepararlo para la carga de nueva información.

Una vez se termine el bucle, el buffer se cerrará y se devolverá el mensaje multiparte creado junto con los adjuntos requeridos.

7 Pruebas

Prueba 1 (P1).

Escenario: Entrar en la aplicación con un usuario y clave para administrador. Comprobar que se muestra el menú correspondiente.

Subprueba 1.1 (S1.1).

Entrar en la aplicación como una asociación. Comprobar que se muestra el menú correspondiente.

Subprueba 1.2 (S1.2).

Entrar en la aplicación con un usuario o clave incorrecta. Comprobar que el sistema detecta que el acceso es erróneo y que muestra por pantalla el error.

Prueba 2 (P2).

Escenario: Una vez dentro de la aplicación como administrador, añadir una nueva asociación en la base de datos. Comprobar que el proceso se ha realizado correctamente, junto con el envío de los correos pertinentes. A continuación e muestran los datos utilizados.

Nombre: Prueba1

Número de póliza: 41235NK324

Efecto: 21-12-2008

CIF: 986986J

Correo electrónico: pruebaproyecto@gmail.com

Calle: Calle Mayor

Número: 2

Piso: 4

Letra: D

Código Postal: 28960

Población: Madrid

Provincia: Madrid

Ilustración 7-1 Prueba añadir asociación

Subprueba 2.1 (S 2.1).

Se introducen campos erróneos, como una fecha de efecto incorrecta o campos obligatorios en blanco.

Prueba 3 (P3).

Escenario: Se validará una asociación una vez recibida la confirmación de la asociación. Comprobar que se manda el correo de validación a la asociación junto con la clave Para ello, se utilizan los siguientes datos:

Persona de contacto: Pedro

DNI: 32147886

Telefono de contacto: 65698789

Correo de contacto: pedroprevigalia@gmail.com

Ilustración 7-2 Prueba validar asociación

Subprueba 3.1 (S 3.1).

Los datos introducidos contendrán algún error, como puede ser un error en el formato del DNI, o dejar vacio algún campo. El sistema debe gestionar estos errores y mostrar por pantalla los campos erróneos.

Prueba 4 (P4).

Escenario: Se procederá a dar de baja una asociación. Comprobar que una vez que se borre la asociación, se borran los voluntarios pertenecientes a ella.

Subprueba 4.1 (S 4.1).

No se marcará ninguna asociación para borrar, por lo que se espera que el sistema una vez se intente validar dicha opción, muestre un mensaje donde aparezca que no se ha marcado ninguna asociación.

Prueba 5 (P5).

Escenario: Modificar los datos de una asociación Comprobar que se modifican correctamente.

Nombre: Prueba 2

CIF: 9867897

Ilustración 7-3 Prueba modificar asociación

Subprueba 5.1 (S 5.1).

Se modifican campos con valores incorrectos. Comprobar que el sistema notifica al usuario los errores cometidos en al inserción de los datos.

Subprueba 5.2 (S 5.2).

Se reenviará la clave de acceso al correo de la persona de contacto dentro de la asociación. Comprobar que le llega el correo junto con la clave.

Prueba 6 (P6).

Escenario: Se procede a dar de alta a voluntarios dentro de una asociación manualmente. Para ello, se rellenan voluntarios de prueba y se seleccionará una asociación donde añadirlos. Comprobar que se dan de alta todos los voluntarios, y se envían los correos correspondientes junto con sus certificados.

Nombre – Primer apellido – Segundo apellido – DNI
V1 – A1 – A1 – 12345698
V2 – A2 – A2 – 2345567
V3 – A3 – A2 – 2349890V

Ilustración 7-4 Prueba añadir voluntarios

Subprueba 6.1 (S 6.1).

Se introducirán campos incorrectos y vacios para dar de alta a los voluntarios. Si existe algún error, el sistema debe detectar antes de validar las incorrecciones, e indicar aquellos voluntarios que contienen fallos en sus campos.

Nombre – Primer apellido – Segundo apellido – DNI
V1 – – A1 – 12345698
V2 – A2 – – 2345567
V3 – A3 – A2 –

Ilustración 7-5 Prueba añadir voluntarios incorrectos

Prueba 7 (P7).

Escenario: Dar de alta a los voluntarios a través de un archivo de texto con un formato determinado. Comprobar que los voluntarios son añadidos correctamente y que se envía el correo correspondiente.

Subprueba 7.1 (S 7.1).

Se da de alta a los voluntarios a través de un archivo. El formato del archivo no es el correcto, y los voluntarios tienen campos que no se corresponden. El sistema debe de sacar por pantalla los voluntarios erróneos junto con los fallos de cada uno.

Prubea 8 (P8).

Escenario: Dar de baja a voluntarios. Comprobar que se borran de la asociación correspondiente y de la base de datos.

Subprueba 8.1 (S 8.1).

Establecer una fecha de baja para los voluntarios. Señalar aquellos voluntarios en los que se desee establecer la fecha de baja, y rellenar en el formato correcto la fecha.

Subprueba 8.2 (S 8.2).

Establecer una fecha de baja errónea. El sistema deberá informar por pantalla al usuario del fallo del fallo de formato en la fecha.

Prueba 9 (P9).

Escenario: Modificar los datos de algún voluntario. Seleccionar una asociación y escoger a un voluntario para modificar sus datos. Comprobar que los cambios se efectúan correctamente.

Subprueba 9.1 (S 9.1).

Modificar los datos de un voluntario por otros incorrectos. El sistema debe informar al usuario de los errores cometidos, y no dejará modificarlos hasta que todos sean correctos.

Prueba 10 (P10).

Escenario: Consultar asociaciones a través de unos parámetros de búsqueda. La salida debe ser un listado de las asociaciones con sus datos correspondientes que coincidan con los criterios buscados. A continuación se muestran los parámetros usados.

Nombre: A

CIF: 2

Provincia: Madrid

Ilustración 7-6 Prueba consultar asociaciones

El sistema debe mostrar todas aquellas asociaciones que comiencen por la letra "a" y su CIF empiece por 2. A su vez, se realiza el filtro por las asociaciones pertenecientes a la Comunidad de Madrid.

Subprueba 10.1 (S 10.1).

No se utilizará ningún parámetro de búsqueda. La aplicación deberá mostrar un listado completo de las asociaciones dadas de alta en el sistema.

Prueba 11 (P11).

Escenario: Se realiza una consulta sobre la base de datos de voluntarios. Se escoge una asociación y se establecen unos criterios e búsqueda para los voluntarios. El sistema debe mostrar un listado de los voluntarios que coincidan con los parámetros, tanto en su comienzo como en su totalidad.

Nombre: Antonio

Fecha alta: 12-02-2003

Ilustración 7-7 Prueba consultar voluntarios

La aplicación mostrará todos los voluntarios que se llamen Antonio y su fecha de alta sea la establecida.

Subprueba 11.1 (S 11.1).

Se seleccionará en la pestaña de asociaciones la opción "Todas". La aplicación deberá mostrar el listado completo de voluntarios que existen en todas las asociaciones.

Modo asociación

Prueba 12 (P12)

Escenario: Modificar los datos pertenecientes a la asociación. Se modifican ciertos parámetros con el fin de comprobar si el proceso de actualización se realiza correctamente. Los parámetros modificados son los siguientes:

CIF: 2349789

Correo: segundapruebaproyecto@gmail.com

Calle: Alfonso XIII

Ilustración 7-8 Prueba modificar asociación

La aplicación modificará los parámetros correspondientes, y mostrará por pantalla si el proceso ha terminado.

Subprueba 12.1. (S 12.1)

Se intentan modificar ciertos parámetros por otros incorrectos. La aplicación deberá informar al usuario de los parámetros erróneos, y no los actualizará hasta que todos ellos estén correctos.

Prueba 13

Escenario: Proceso de alta de los voluntarios para la asociación. Se rellenan diferentes voluntarios para darlos de alta en la aplicación. El sistema deberá generar los correos y certificados correspondientes, y mostrar por pantalla los voluntarios que hallan sido correctamente almacenados.

Subprueba 13.1 (S 13.1)

Se dará de alta a voluntarios con formato de campos incorrecto. La aplicación deberá detectar estos fallos, e informar al usuario de ello.

Prueba 14 (P14)

Escenario: Proceso de búsqueda de voluntarios a partir de los parámetros deseados. La aplicación deberá mostrar un listado de los voluntarios que coincidan con los parámetros de búsqueda. A su vez, debe permitir generar un libro de registro en formato PDF del listado obtenido. Los parámetros usados son estos:

Nombre: P **Primer apellido**: Gonzalez **Fecha de alta**: 12-01-2003

Ilustración 7-9 Prueba consultar voluntarios

El sistema mostrará por pantalla un listado que contenga a los voluntarios cuyo nombre comience por "P", su primer apellido sea "González" y la fecha de alta sea el 12 de enero de 2003.

Subprueba 14.1 (S 14.1)

Se realizará una búsqueda sobre los voluntarios sin criterio de búsqueda. El sistema debe generar un listado con todos los voluntarios pertenecientes a la asociación que lo consulta.

Resultado de las pruebas

P1	☑

S 1.1	☑
S 1.2	☑
P2	☑
S 2.1	☑
P3	☑
S 3.1	☑
P4	☑
S 4.1	☑
P5	☑
S 5.1	☑
S 5.2	☑
P6	☑
S 6.1	☑

P7	☑
S 7.1	☑
P8	☑
S 8.1	☑
S 8.2	☑
P9	☑
S 9.1	☑
P10	☑
S 10.1	☑
P11	☑
S 11.1	☑
P12	☑
S 12.1	☑

P13	☑
S 13.1	☑
P14	☑
S 14.1	☑

Tabla 36. Resultados de las pruebas

8 Conclusiones

En este apartado se detallan los problemas que han surgido en el desarrollo de la aplicación, junto con los objetivos alcanzados y conclusiones finales.

8.1 Problemas encontrados al realizar el proyecto

Durante el tiempo que ha durado la realización del proyecto, han surgido numerosos problemas, gran parte propiciados por la inexperiencia en un desarrollo de este tipo. A continuación se exponen los más destacados junto con su solución.

- El primer problema que surgió, fue el cambio de proyecto repentinamente. Siguiendo el mismo tipo de aplicación Web del proyecto anterior, se tuvo que cambiar un desarrollo ya avanzado por otro completamente nuevo, aunque se pudo reutilizar gran parte del anterior. Los cambios retrasaron el tiempo estimado, hubo que recalcular todo de nuevo, pero se consideró que el cambio sería beneficioso.

  Finalmente y tras acabar el proyecto, se puede decir que se acertó al desarrollar esta aplicación, se le ha dado un uso que seguramente la otra no hubiera tenido.

- Una vez decidida la aplicación a realizar, el siguiente problema fue qué lenguajes utilizar. Previamente, para el otro proyecto se utilizaron herramientas como Java, Jsp o MySQL, y se debía tomar la decisión de optar por estas o plantearse el uso de otras nuevas. Debido a los conocimientos adquiridos anteriormente, se decidió optar por las herramientas del otro proyecto, y sin lugar a dudas, fue un acierto.

  Como se puede ver en apartados anteriores, el software utilizado es uno de los más adecuados para el desarrollo Web, permitiendo gran flexibilidad y reutilización a la hora de programar. Disponen de una amplia documentación vía Internet, permitiendo un aprendizaje rápido, a través de ejemplos y tutoriales.

El uso de Java en el desarrollo, permitió usar varias de sus librerías gratuitas, entre las que se encuentran Itext o JavaMail. Supuso un incremento en el tiempo de aprendizaje, aunque una vez dominadas, se pudo ver la potencia de la que disponían, permitiendo acortar el tiempo de programación de la aplicación.

- Uno de los principales problemas que ha tenido el proyecto, ha sido el cambio constante por parte del cliente de los requisitos de la aplicación.

  Se realizó un primer análisis de requisitos, especificado claramente por el cliente, pero a medida que el desarrollo iba avanzando, los requisitos fueron cambiando. Hubo que modificar gran parte de la aplicación, ya que o bien el cliente no lo había pensado bien en un principio, o bien al ver el resultado final se dio cuenta de los fallos que había cometido. En todo momento se intento respetar el diseño y los requisitos que el cliente había elegido.

  Todas estas modificaciones no hicieron más que retrasar el tiempo de desarrollo.

  Finalmente, la aplicación se dio por terminada, y actualmente, esta en periodo de mantenimiento.

- Por último, destacar la laboriosa tarea que es instalar y configurar las aplicaciones necesarias para desarrollar el proyecto. Muchas de ellas si se configuran mal desde un principio, guardan los valores de instalación cuando se desinstalan, por lo que hay que borrar ciertos archivos de configuración para volver a empezar de cero.

  Sincronizarlas con el NetBeans supuso también una gran cantidad de tiempo, e incluso una vez aprendido el método, sigue siendo una tarea nada fácil.

8.2 Objetivos

El principal objetivo de este proyecto fin de carrera, era crear una aplicación que permitiese gestionar los seguros de los voluntarios pertenecientes a las distintas asociaciones que tienen contratadas sus pólizas a través de la empresa administradora. El sistema debía generar automáticamente los certificados de cada asegurado, y emitirlo a la asociación

correspondiente. Solo esta tarea, ya significaba un gran ahorro en tiempo y costes para la empresa administradora.

Se podría asegurar que el principal objetivo del desarrollo se ha cumplido, pese al gran trabajo que ha supuesto, y que debido a factores tanto internos como externos, el tiempo estimado para el desarrollo del proyecto ha sido mayor de lo esperado.

A continuación se exponen los objetivos más destacados.

1.- Realizar una herramienta que permita a la empresa administradora gestionar las altas, bajas y modificaciones de los asegurados.

2.- Almacenar en una base de datos externa a la empresa administradora, los datos de las asociaciones y asegurados, con el fin de tener en otro soporte de almacenamiento toda la información de la que disponen.

3.- El sistema deberá contar con una interfaz sencilla e intuitiva, que permita a los usuarios utilizarla sin ninguna dificultad.

4.- La aplicación debe ser capaz de generar todos los documentos pertinentes, contratos, certificados, listados, etc., y enviarlos o mostrarlos por pantalla según proceda en cada tarea.

5.- El sistema debe de gestionar cualquier error o suceso inesperado que se registren al uso de la aplicación. Deberá informar al usuario e impedir que el proceso se realice cuando se produzca un fallo, ya sea causado por el usuario o por la aplicación.

6.- Diseñar una aplicación donde su mayor parte sea reutilizable en un futuro, bien para rediseñar una nueva y poder usarla como base para su realización, o bien para modificarla y que sirva para otras empresas que se dedican al mismo sector.

8.3 Conclusiones finales

Destacar en primer lugar el aprendizaje tanto personal como profesional que ha supuesto la realización de este proyecto.

Se han adquirido numerosos conocimientos en diversas ramas de la informática, hoy en día usadas por muchísimas empresas para el desarrollo de software. Entre ellas, cabe destacar las siguientes:

- Ingeniería del software.
- Desarrollo de aplicaciones Web.
- Diseño y desarrollo de bases datos sobre MySQL.
- Desarrollo en Java.
- Diseño de la interfaz en Jsp.

A su vez, se han desarrollado cualidades y aptitudes tanto personales como profesionales:

- Creatividad.
- Responsabilidad.
- Aprender a ser autodidacta en algunos aprendizajes.
- Aptitud para la resolución de problemas.
- Constancia.
- Capacidad de organización.
- Capacidad de adaptación.
- Paciencia.

9 Trabajos futuros

Tanto al comienzo como a medida que se iba desarrollando el proyecto, fueron surgiendo varias ideas y mejoras que a día de hoy aún no se han realizado. A continuación se exponen las que se han considerado más interesantes y factibles en un futuro.

- Creación de una nueva interfaz para el administrador, donde poder modificar toda la aplicación.

  Para ello, se deberían programar más funcionalidades de las que ahora tiene la aplicación, para que el administrador desde su menú, pueda activar o desactivar aquellas que considere necesarias dependiendo de la etapa en la que se encuentre la gestión.

- Alta de asociaciones realizada por ellas mismas.

  Actualmente, en la aplicación solo pueden darse de alta asociaciones a través del administrador. En un futuro se podría plantear el alta por parte de las asociaciones, que fuesen ellas mismas quienes rellenen sus propios datos, y que el administrador tan solo se encargase de validar el alta.

- Dar opción en las tablas y listados de ordenarlos por el campo que el usuario desee.

  Esta requisito no ha sido exigido, ya que al administrador tan solo le interesa que los listados estén ordenados a través del campo "Primer apellido", sin embargo, sería una buena mejora implementar esta posibilidad, que, aunque en un principio no es imprescindible, si que puede facilitar búsquedas en algunos momentos.

- Ofrecer al asegurado la posibilidad de entrar en la aplicación a consultar sus datos y fecha de vencimiento del seguro

Incluir un nuevo usuario en la aplicación, el asegurado. A través de un nombre y una clave generada dinámicamente para cada uno, ofrecer la posibilidad de consultar sus datos y fechas del seguro.

- Optimizar el acceso a la aplicación mediante un Pull de conexiones.

  Actualmente, las conexiones por parte de los usuarios a la aplicación se esta gestionando de una manera efectiva, creando y cerrando conexiones cuando se este usando la base de datos. Sin embargo, una mejora mas eficiente sería crear un Pull de conexiones, lo que dotaría al sistema de una rapidez mucho mayor.

- Dotar a la aplicación de una mejor interfaz.

  Se podría suprimir el diseño con frames, y optar por un diseño más vistoso e intuitivo. Aunque el que se ha realizado cumple perfectamente con todo lo requerido, siempre queda un cierto grado de mejora que en este caso podría realizarse sin ningún problema.

- Ampliar la aplicación ofreciendo la posibilidad de gestionar seguros aéreos.

  Esta fue una de las primeras mejoras que se planteo, ya que el cliente tenía en mente estas dos aplicaciones. Habría que crear una nueva aplicación, o modificar gran parte de la interfaz de esta para poder gestionar las dos a la vez. Esta mejora muy posiblemente acabe realizándose, puesto que, al igual que la aplicación desarrollada, reducirá el tiempo de gestión de los asegurados por parte de la empresa administradora.

Apéndice A. Manual de instalación

A.1. Hardware y software necesario.

A.1.1. Hardware

* El hardware necesario para instalar en la aplicación dependerá de qué medio usemos para la instalación. La aplicación permite instalarse tanto en un ordenador local, instalando el software necesario para simular un servidor, o en un servidor que permita el acceso a los demás usuarios. En un principio, la aplicación esta dirigida para instalarla en un servidor Web y poder usarla a través de Internet. Por lo tanto a continuación se explican los distintos requisitos para las diferentes instalaciones.

A.1.1.1 Instalación en un ordenador personal

* Procesador Intel Core Duo 1.06 GHz ó superior.

* Un mínimo de 512 Mb de memoria RAM.

* 20 Mb de disco duro para la instalación de la aplicación. Si no se dispone las utilidades necesarias para poder utilizarla, se requerirán 280 Mb de disco duro.

A.1.1.2 Instalación en servidor Web

* Procesador Intel Core Duo 2 Ghz ó superior.

* Un mínimo de 1 Gb de memoria RAM.

* 20 Mb de disco duro para la instalación de la aplicación.

A.1.2. Software

* Referente al software, también se dispone de dos posibilidades, dependiendo si la instalación se realizará sobre un servidor o sobre un ordenador personal.

A.1.2.1 Instalación en un ordenador personal

- MySQL 5.1.30
- Apache HTTPD 2.2.11
- PhpMyAdmin 3.1.1
- Navegador
- Sistema operativo Windows Xp o superior.

A.1.2.2 Instalación en un servidor Web

* Apache Tomcat 5.5 o superior con Java

* MySQL 5 o superior.

* MySQL WebAdmin

* Sistema operativo Windows Server o Linux.

A.2. Instalación de la aplicación.

A.2.1 Instalación en un ordenador personal

Para la instalación de la aplicación en un ordenador personal se debe instalar el software correspondiente si este no esta instalado previamente en la máquina. Para ello, se puede optar por dos opciones, una muy sencilla, instalando un paquete que contiene todos los elementos necesarios para su ejecución, o bien, una más complicada instalando uno a uno los componentes.

Se explicará con imágenes y sencillos pasos la instalación del paquete entero, dejando libertad al usuario para la instalación componente por componente.

- Descarga del software necesario a través de la siguiente página.

http://www.apachefriends.org/es/xampp.html

- Se mostrará la siguiente pantalla, donde debemos elegir el sistema operativo donde queremos instalar el software.

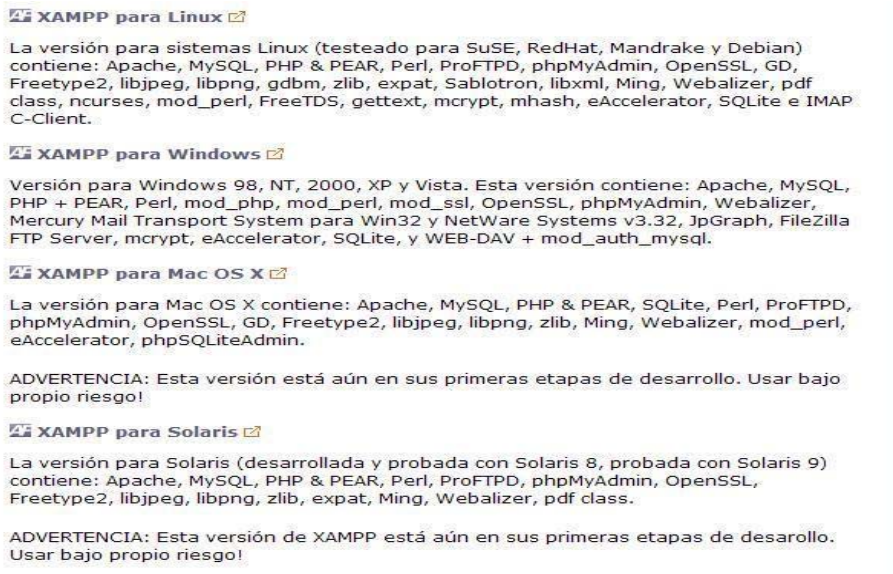

Ilustración 8-1 SO para Xampp

- Una vez seleccionado el sistema operativo, se muestra la pantalla de descargas, donde se debe elegir el archivo "Installer" en la opción XAMPP Windows 1.7.0.

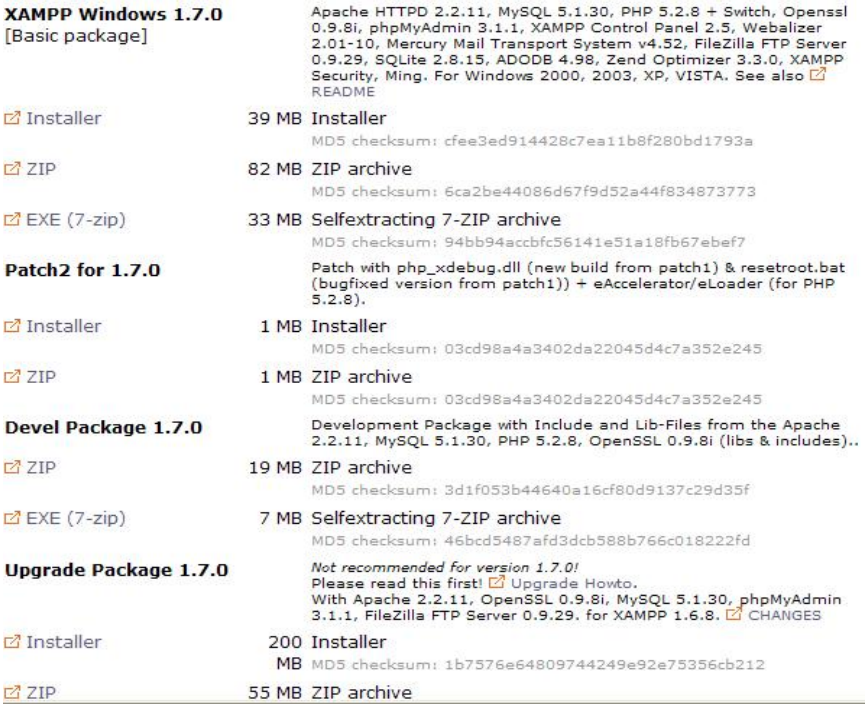

Ilustración 8-2 Descargar Xampp

- Se descargará el archivo en la ruta indicada. Doble click en la aplicación, y comenzará el proceso de instalación.

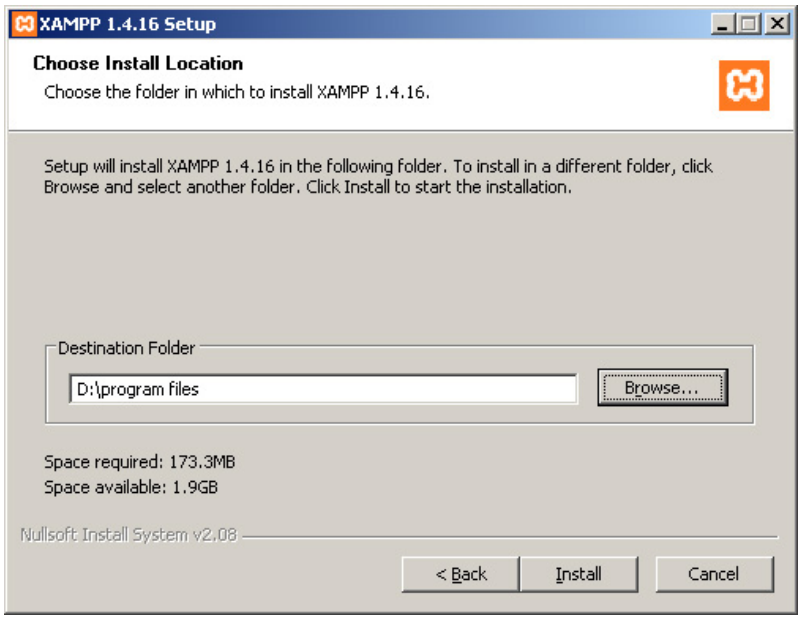

Ilustración 8-3 Proceso Instalción Xampp

- Siga las instrucciones de la instalación hasta completarla. Una vez instalado, ejecuta Xampp si no se ha ejecutado automáticamente.

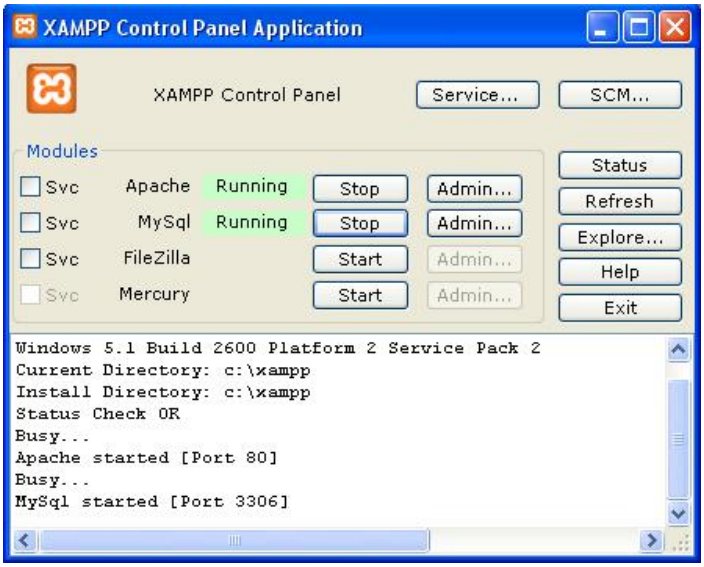

Ilustración 8-4 Ejecución Xampp

- Escriba en su navegador http://127.0.0.1. Aparecerá la siguiente pantalla de bienvenida, donde deberá escoger el lenguaje a utilizar.

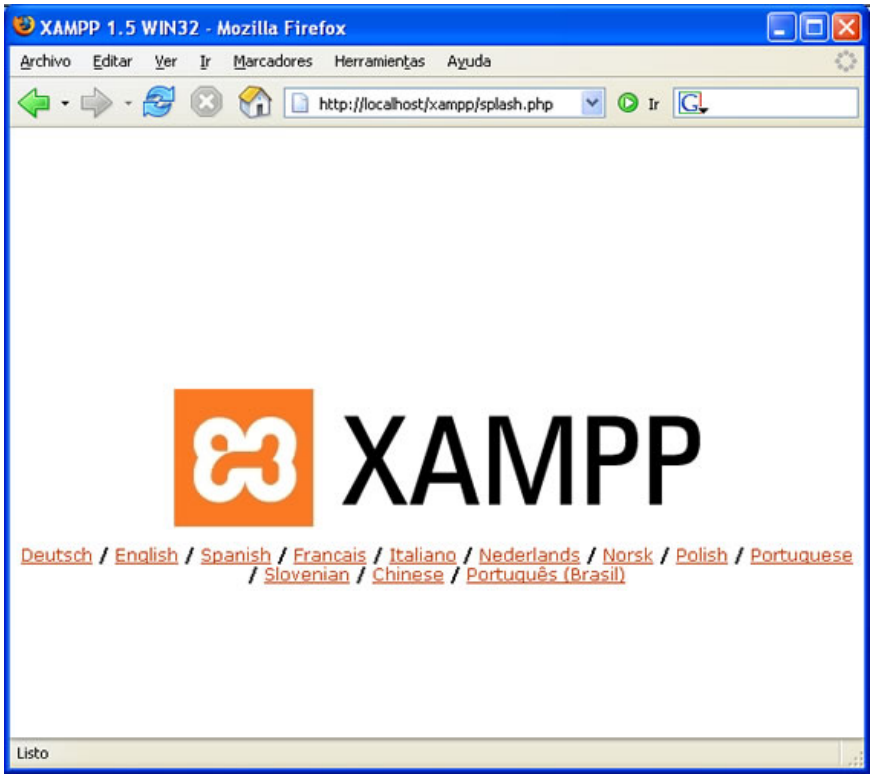

Ilustración 8-5 Ventana principal navegación Xampp

- Una vez escogido el idioma, se mostrará la siguiente pantalla, el panel de control, desde donde gestionar Xampp.

Ilustración 8-6 Menú navegación Xampp

- Se debe crear una nueva base de datos. Para ello, diríjase a "phpMyAdmin", donde podrá seleccionar dicha opción. El nombre de la base de datos será "previgalia".

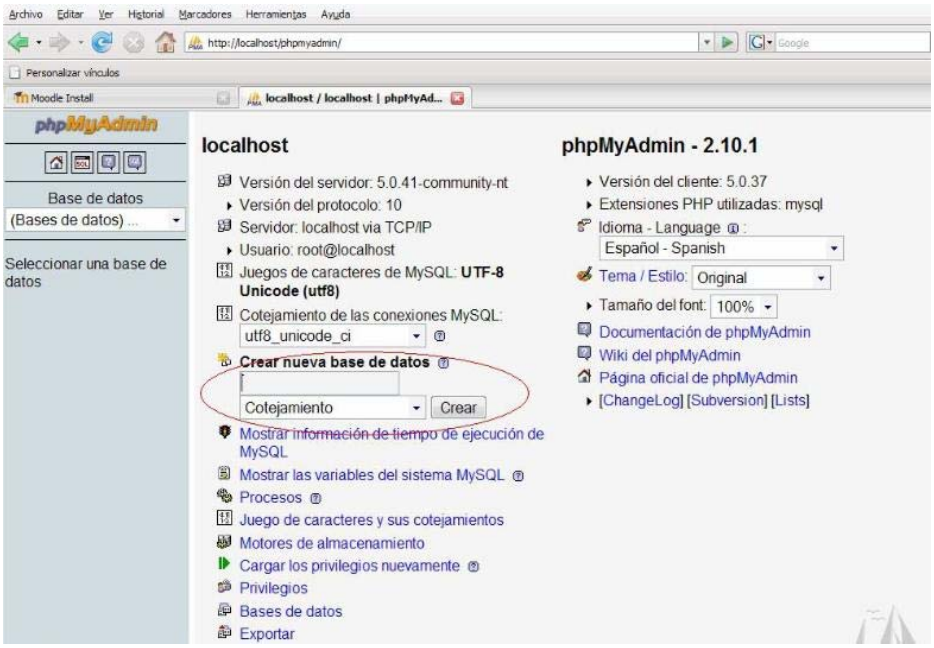

Ilustración 8-7 Crear una base de datos Xampp

- Una vez creada la base de datos, importamos el archivo que contiene los parámetros necesarios.

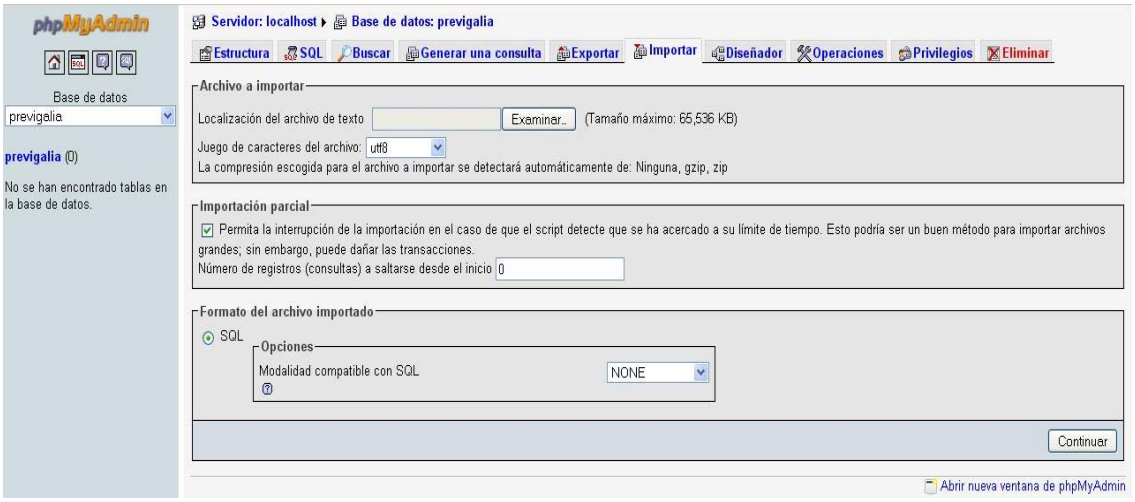

Ilustración 8-8 Importar archivo en Xampp

- Terminadas estas operaciones, ya esta montada la base de datos. Si todo ha salido correctamente, debe quedar una estructura igual que la de la imagen inferior.

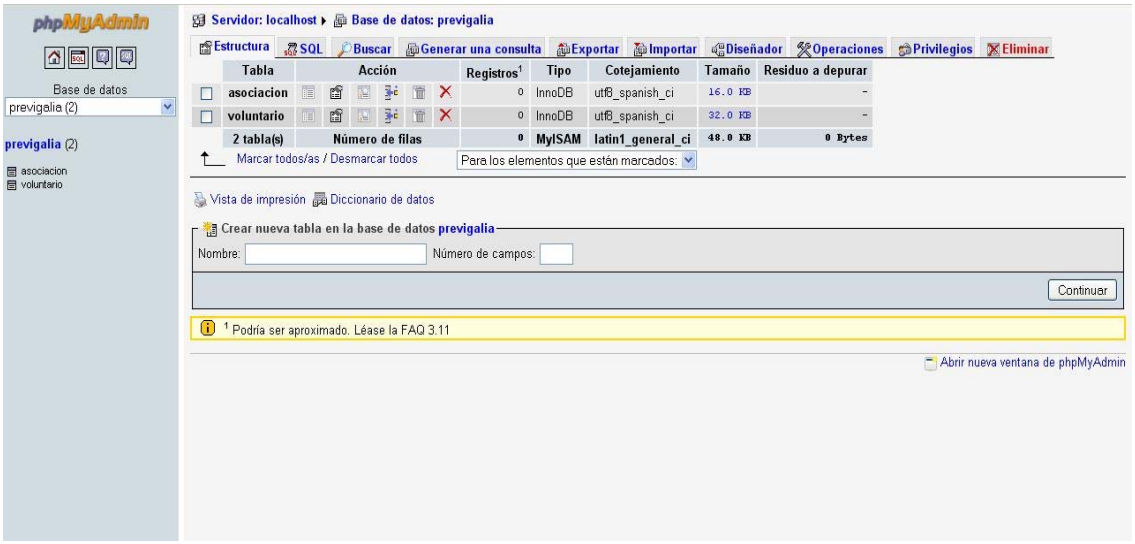

Ilustración 8-9 Base de datosXampp

- Cree un directorio dentro de "Mis Documentos", llamado "NetBeansProyects", y dentro de este otro llamado "Previgalia", que será donde debe almacenar todo el contenido de la aplicación, dejando la estructura interna de la aplicación tal y como esta.

```
/build/

build.xml

/dist/

/nbproyect/

/src/

/test/

/web/
```

Ilustración 8-10 Directorios servidor

- Si la instalación ha sido satisfactoria y ha seguido los pasos correctamente, podrá accedes a la aplicación a través de su navegador mediante la siguiente dirección:

  http://localhost:8080/Previgalia/

A.2.2 Instalación en un servidor web

- Si ya ha escogido el servidor web donde alojar la aplicación, deberá seguir los siguientes pasos para instalarla.
- Siga los pasos de la instalación en ordenador personal sobre la creación de la base de datos. Para ello, deberá acceder al panel de control que le ofrezca su servidor web.

- Una vez creada, deberá acceder al FTP de su página, donde dispondrá de una estructura de directorios parecida a esta.

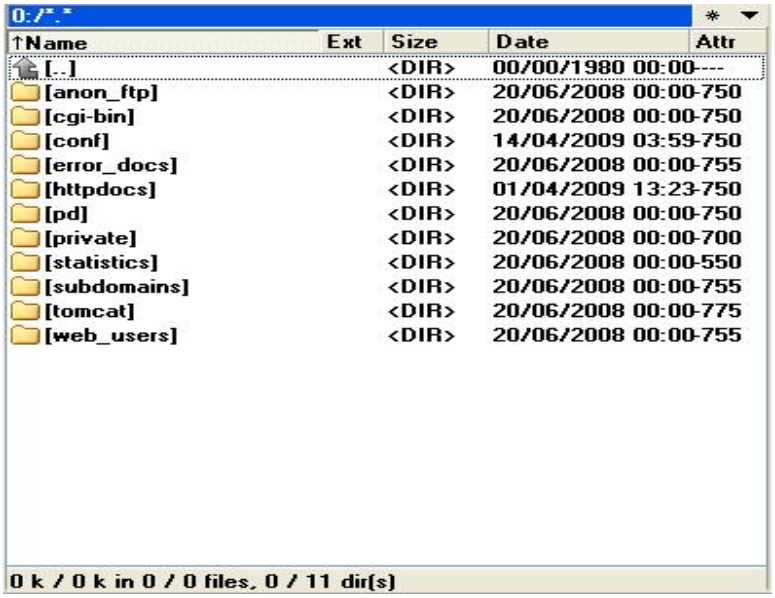

Ilustración 8-11 Estructura directorios ftp

- Diríjase al directorio "httpdocs", donde deberá copiar las carpetas que contiene la aplicación.
- Si estas carpetas no coincidiesen, la estructura que debe tener es la siguiente:

/httpdocs/

/httpdocs/images/

/httpdocs/WEB-INF/classes/com/previgalia/

Ilustración 8-12 Estructura directorios 2 ftp

  - En el directorio principal, deberán ir los archivos con formato .jsp.
  - El directorio images contendrá las imágenes de la aplicación.
  - El último directorio, será el que contenga los archivos compilados.

- Finalmente, para acceder a la aplicación, deberá ingresar en el dominio que contrató.

Apéndice B. Manual de usuario.

Consideraciones generales.

En este aparatado se describen consideraciones que todo usuario debe tener en cuenta al usar la aplicación.

- El estado de la sesión pasará a inactivo si transcurren 30 minutos del último uso de sesión.

- Todo usuario deberá desconectar la sesión una vez haya acabado de realizar las gestionar pertinentes.

- Siempre que se desee, se puede consultar la ayuda en el menú principal, si hubiese algún problema que con la ayuda no pudiese resolver, podrá ponerse en contacto con el administrador en la dirección de correo que se facilita en la parte superior derecha.

- La aplicación esta testada en diversos navegadores, si por algún casual utiliza uno en el que la aplicación no se visualizase correctamente, póngase en contacto con el administrador.

- Tenga especial atención en la inserción de datos, cualquier error que cometa le será informado por pantalla para que puede resolverlo.

- Si no le llegasen los correos que el sistema envía al realizar diversas acciones, mire en primer lugar en su buzón de correo en la carpeta "Correo no deseado". Si aún así no lo ha recibido, póngase en contacto con el administrador para que este lo pueda volver a enviar.

- Algunos ficheros que el sistema genera tienen formato PDF, para visualizarlos correctamente, deberá tener instalado Adobe Acrobat Reader.

Página de bienvenida

Pantalla de bienvenida para todos los usuarios del sistema. Para acceder se requiere un nombre de usuario y una contraseña, pulsando posteriormente sobre Iniciar sesión.

Ilustración 9-1 Página Bienvenida aplicación

Contraseña o usuario incorrecto

Si el usuario o la contraseña es incorrecta, se mostrará este mensaje de error, teniendo que volver a introducir el usuario o la contraseña correctamente.

Ilustración 9-2 Insercción datos aplicación

Modo Administrador

Menú administrador

Si el acceso ha sido correcto por parte del administrador, se le mostrará esta pantalla, donde tendrá acceso a todas las opciones del menú.

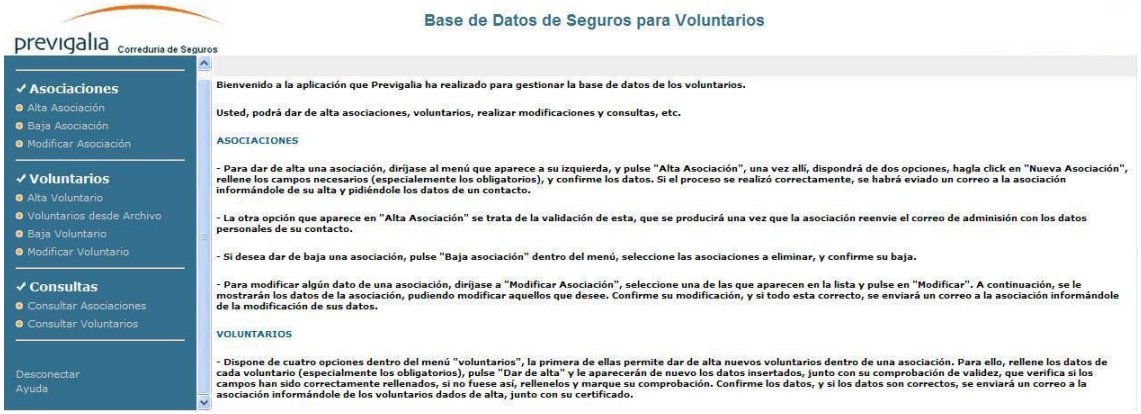

Ilustración 9-3 Pantalla principal administrador

Gestionar Asociaciones

Alta Asociación

Pantalla que permite al administrador comenzar el proceso de alta de una nueva asociación. Para ello, en primer lugar deberá acceder a la opción Nueva Asociación, donde introducirá los datos correspondientes a esta. Una vez introducidos y validados por la asociación, el administrador validará los nuevos datos y dará por completado el alta de la nueva asociación.

Ilustración 9-4 Pantalla Alta asociación administrador

Nueva Asociación

El administrador deberá rellenar los datos de la asociación, siendo obligatorios aquellos campos que lo soliciten. Si algún dato es incorrecto se mostrará por pantalla el error oportuno. Una vez completados los campos, deberá pulsar dar de alta, para posteriormente comprobar los datos introducidos y confirmarlos.

Ilustración 9-5 Pantalla modificar asociación administrador

Validar Asociación

Una vez completado el proceso anterior, la asociación recibirá la petición de alta junto con los datos introducidos. Si todo esta correcto, informará de ello al administrador para que

este valide el proceso y complete le información que falta. Una vez validada, se envian el correspondiente usuario y contraseña a la asociación.

Esta primera pantalla muestra al administrador las asociaciones disponibles para su validación.

Ilustración 9-6 Pantalla selección asociación a validar

Esta segunda pantalla, muestra la asociación escogida junto con los campos a rellenar.

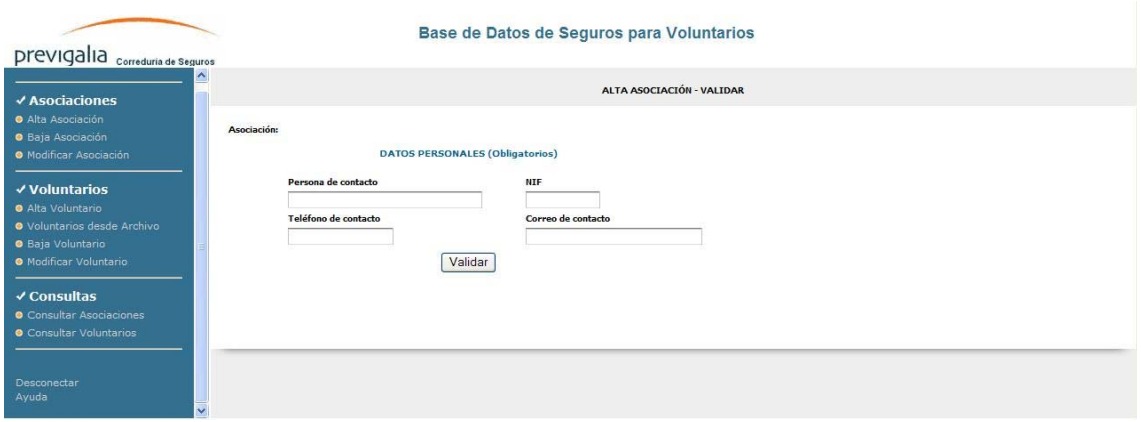

Ilustración 9-7 Pantalla validación asociación

Baja Asociación

El administrador puede dar de baja a las asociaciones siempre y cuando estas lo hayan requerido, o bien la asociación haya desaparecido. En cualquier caso, el sistema mostrará al administrador una lista con las asociaciones disponibles, para que elija aquellas que tengan que ser dadas de baja.

Ilustración 9-8 Pantalla baja asociación

Modificar Asociación

Si por algún motivo el administrador tuviese que modificar algún dato de una asociación, se le mostrará una lista con las asociaciones disponibles, de la que deberá escoger aquella que desee modificar. Una vez elegida, el sistema mostrará en pantalla todos los datos disponibles de la asociación. El administrador modificará aquellos que estén incorrectos.

Esta será la lista de asociaciones que verá el administrador.

Ilustración 9-9 Pantalla elección asociación a modificar

Una vez elegida, se muestran los datos. Si fuese necesario, se podrá reenviar desde esta pantalla, la contraseña a la asociación, siempre bajo petición de esta. Para ello tan solo deberá pulsar sobre el botón Reenviar Clave.

Ilustración 9-10 Pantalla modificar asociación administrador

Gestionar Voluntarios

Alta Voluntario

Para dar de alta a un voluntario, el administrador tendrá que rellanar los campos obligatorios para cada voluntario, seleccionando una vez haya terminado, la asociación en la que tienen que ser dados de alta. El sistema comprobará que no falta ningún dato, teniendo que confirmar el administrador dicha comprobación.

Pantalla de alta de voluntarios.

Ilustración 9-11 Pantalla alta voluntarios administrador

Pantalla de confirmación de los voluntarios.

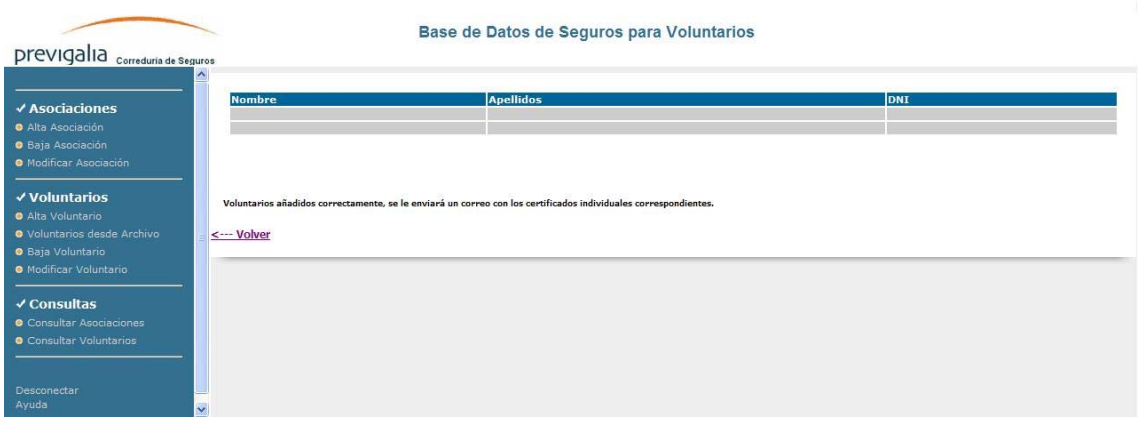

Ilustración 9-12 Pantalla confirmación alta voluntarios administrador

Alta de voluntarios desde archivo

Para simplificar y acelerar el proceso de alta, se permite al administrador la opción de introducir en el sistema a voluntarios a través de un fichero de texto con un formato determinado.

Ilustración 9-13 Pantalla alta voluntarios desde archivo

Si hubiese algún error en los voluntarios que hay en el archivo, se mostrará la siguiente pantalla donde se informará de los errores que tiene cada voluntario.

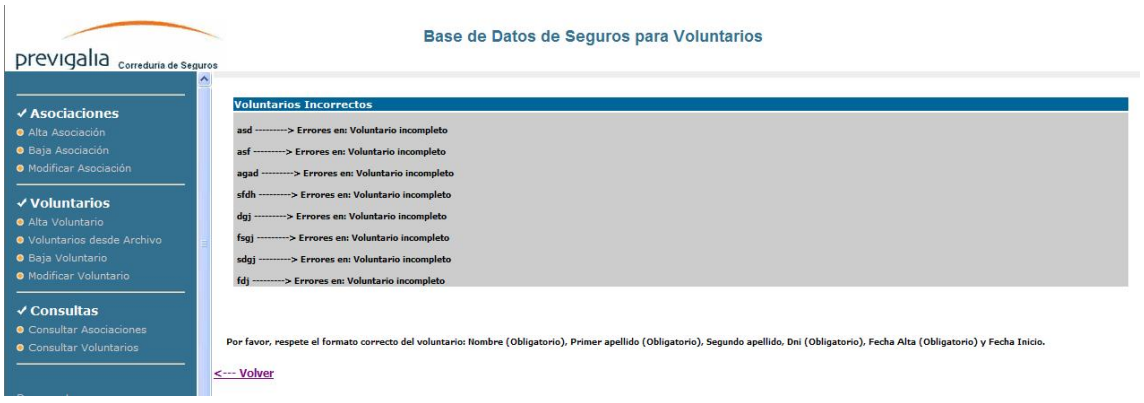

Ilustración 9-14 Pantalla error alta voluntario desde archivo

Baja voluntario

Se muestra al administrador una lista de las asociaciones que contienen voluntarios a dar de baja. Este, selecciona una de ellas, a continuación, el sistema carga en pantalla los datos de los voluntarios pertenecientes a la asociación escogida. El administrador puede hacer dos acciones, o bien eliminar al voluntario de la base de datos, o bien establecer una fecha de baja para cada uno de los voluntarios seleccionados.

Pantalla de selección de asociación.

Ilustración 9-15 Pantalla selección asociación baja de voluntario

Pantalla de baja de voluntarios.

Ilustración 9-16 Pantalla baja de voluntarios

Modificar voluntario

Pantalla que permite la modificación de un voluntario. El administrador deberá elegir la asociación donde este el voluntario a modificar. Una vez elegida, deberá seleccionar el voluntario, y una vez seleccionado, el sistema mostrará los datos de este para que puedan ser modificados.

Pantalla selección asociación.

Ilustración 9-17 Pantalla elección asociación modificar voluntarios

Pantalla selección voluntario.

Ilustración 9-18 Pantalla selección voluntario modificar

Pantalla datos del voluntario.

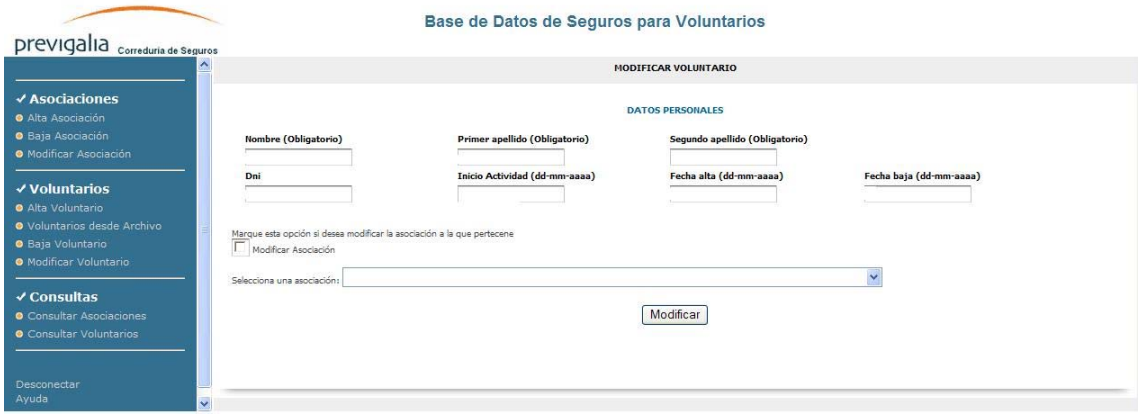

Ilustración 9-19 Pantalla modificar voluntario

Consultas

Consultar Asociaciones

Si el administrador desea realizar una consulta sobre las asociaciones disponibles, deberá rellanar aquellos campos por los cuales quiere que se realice el criterio de búsqueda.

Pantalla de consulta de asociaciones.

Ilustración 9-20 Pantalla consultar asociaciones

Pantalla de lista de asociaciones.

Ilustración 9-21 Pantalla asociaciones consultadas

Consultar voluntarios

Permite realizar una búsqueda sobre los voluntarios introducidos en la base de datos. A su vez, el administrador puede solicitar un libro de registro sobre los voluntarios seleccionados presionando sobre el botón imprimir.

Pantalla de consulta de voluntarios.

Ilustración 9-22 Pantalla consultar voluntarios

Pantalla de lista de voluntarios.

Ilustración 9-23 Pantalla voluntarios consultados

Modo Asociación

Menú Asociación

Si el acceso ha sido correcto por parte de la asociación, el sistema presentara en pantalla el siguiente menú.

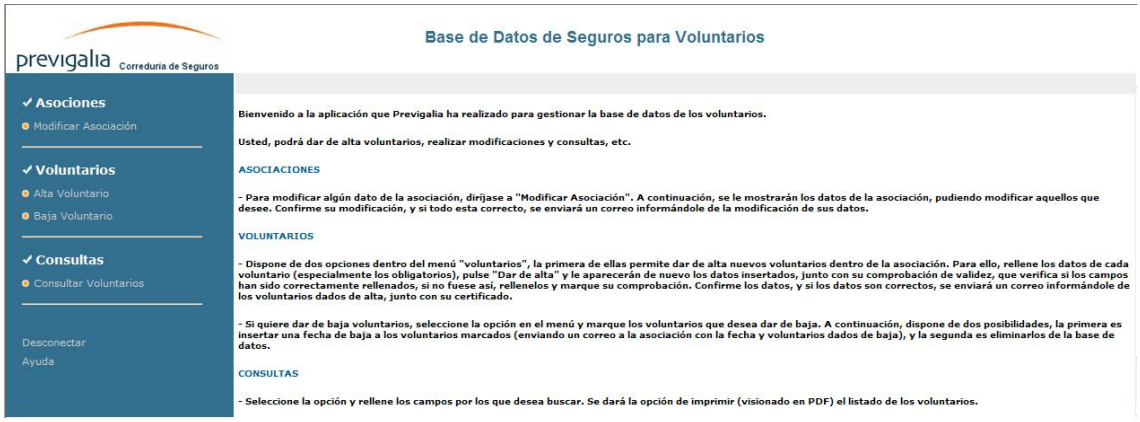

Ilustración 9-24 Pantalla principal asociación

Gestionar Asociación

Modificar asociación

Si la asociación lo desea, podrá modificar sus datos en cualquier momento. Se mostrará por pantalla la información que el sistema tiene sobre esta. Si se realiza alguna modificación y esta es incorrecta, el sistema informa al usuario del dato erróneo.

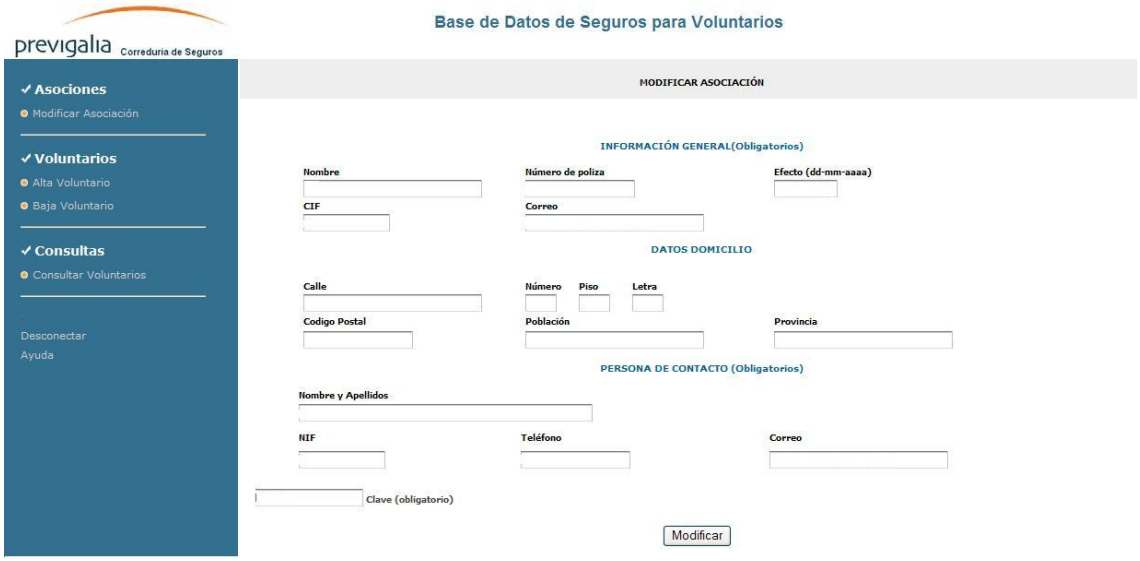

Ilustración 9-25 Pantalla modificar asociación de asociación

Gestionar Voluntarios

Alta voluntario

El usuario deberé rellenar los campos obligatorios de los voluntarios a dar de alta. Si hubiese algún error, se informará por pantalla de los datos incorrectos.

Pantalla alta voluntario.

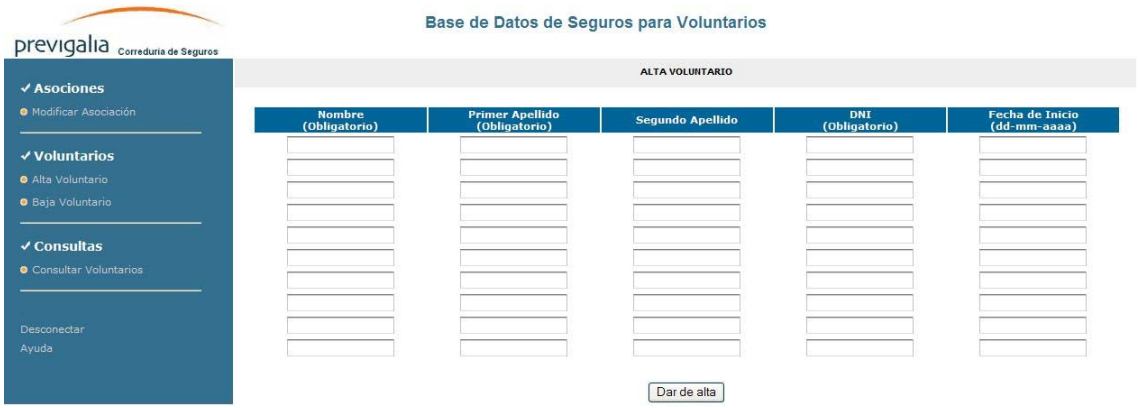

Ilustración 9-26 Pantalla alta voluntario asociación

Pantalla de confirmación alta de voluntarios.

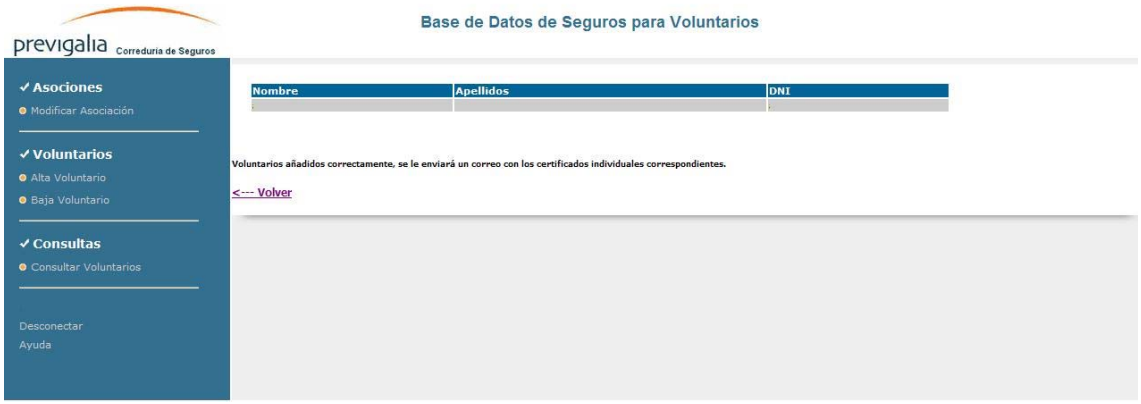

Ilustración 9-27 Pantalla confirmación alta voluntario asociación

Baja voluntario

Pantalla que permite o bien eliminar a los voluntarios de la base de datos, o bien establecer una fecha de baja para los voluntarios seleccionados.

Ilustración 9-28 Pantalla baja voluntario asociación

Consultas

Consultar voluntarios

El usuario podrá realizar búsquedas sobre la base de datos de voluntarios que contenga su asociación. A su vez, y una vez realizada la búsqueda, podrá imprimir por pantalla en formato de libro de registro su selección.

Pantalla de búsqueda.

Ilustración 9-29 Pantalla consultar voluntarios asociación

Pantalla que muestra los datos de los voluntarios.

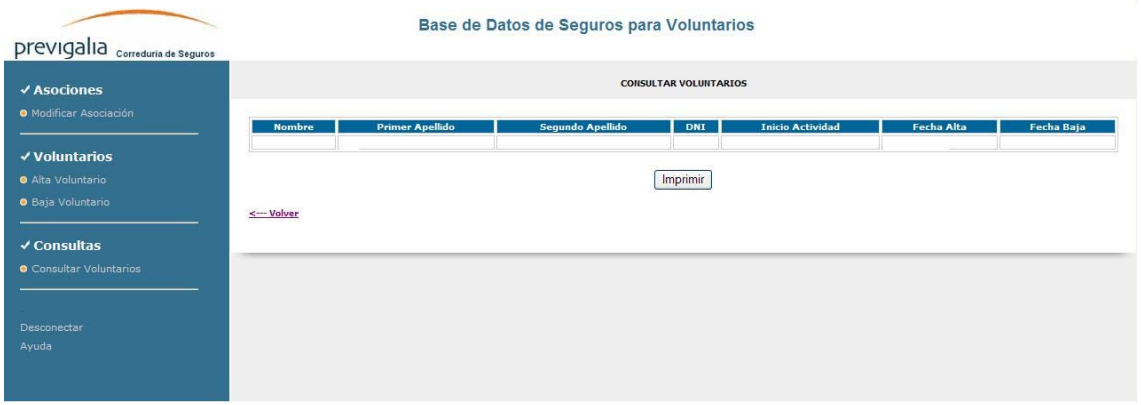

Ilustración 9-30 Pantalla voluntarios consultados asociación

Anexo I

	Nombre de tarea	Duración	Comienzo	Fin
1	INICIO DEL PROYECTO	1 día?	lun 23/02/09	lun 23/02/09
2	**ACTIVIDADES PREVIAS**	**3 días**	**lun 23/02/09**	**mié 25/02/09**
3	Investigación	1 día	lun 23/02/09	lun 23/02/09
4	Elegir Ciclo de vida	1 día	mar 24/02/09	mar 24/02/09
5	Crear proceso de Ciclo de Vida	1 día	mié 25/02/09	mié 25/02/09
6	Fin inicio poryecto	0 días	mié 25/02/09	mié 25/02/09
7	**GESTIÓN**	**6,01 días**	**jue 26/02/09**	**vie 06/03/09**
8	**PLANIFICACIÓN U ORGANIZACIÓN**	**2 días**	**jue 26/02/09**	**vie 27/02/09**
9	Planificar Gestión Proyecto	2 días	jue 26/02/09	vie 27/02/09
10	**ESTIMACIÓN**	**4,01 días**	**lun 02/03/09**	**vie 06/03/09**
11	Identificar recursos	2,01 días	lun 02/03/09	mié 04/03/09
12	Ejecutar Estimación	2 días	mié 04/03/09	vie 06/03/09
13	Fin Gestión	0 días	vie 06/03/09	vie 06/03/09
14	**ANÁLISIS**	**18,03 días**	**vie 06/03/09**	**mié 01/04/09**
23	Fin Análisis	0 días	mié 01/04/09	mié 01/04/09
24	**DISEÑO**	**25 días**	**mié 01/04/09**	**mié 06/05/09**
32	Fin de diseño	0 días	mié 06/05/09	mié 06/05/09
33	Formación del personal	8 días	mar 07/04/09	jue 16/04/09
34	**IMPLEMENTACIÓN**	**19 días**	**mié 06/05/09**	**mar 02/06/09**
42	Fin implementación	0 días	mar 02/06/09	mar 02/06/09
43	**PRUEBAS**	**7 días**	**mar 02/06/09**	**jue 11/06/09**
47	Fin Pruebas	0 días	jue 11/06/09	jue 11/06/09
48	Instalación del Software	1 día	jue 11/06/09	vie 12/06/09
49	Aceptación del Software	1 día	vie 12/06/09	lun 15/06/09
50	Producir y distribuir documentación	1 día	lun 15/06/09	mar 16/06/09
51	**SEGUIMIENTO**	**73 días**	**lun 23/02/09**	**mié 03/06/09**

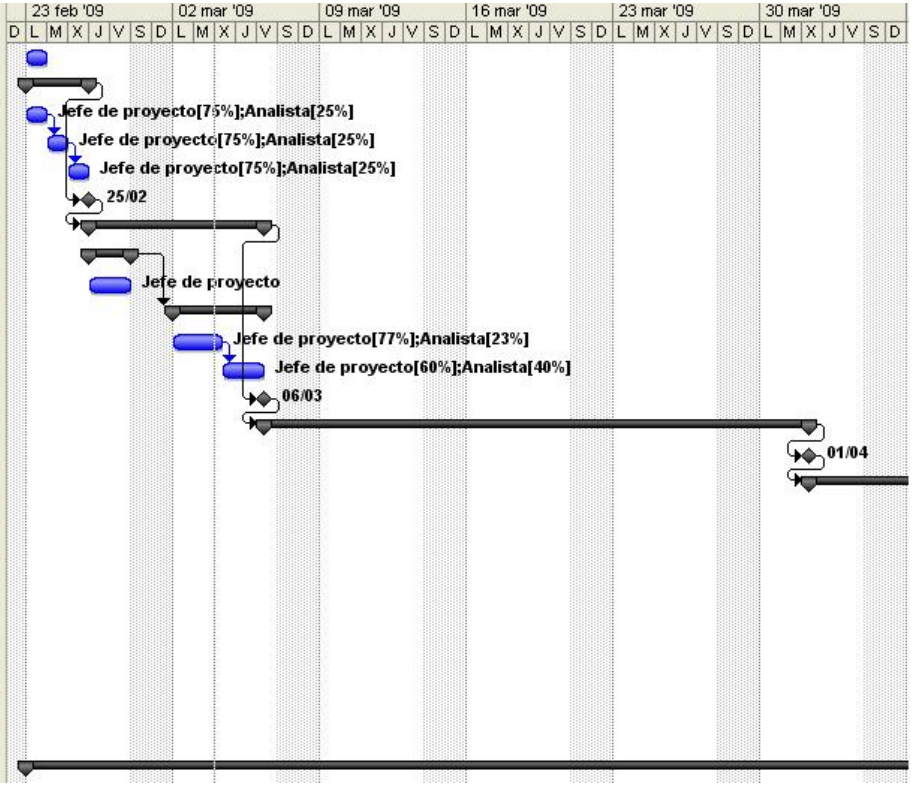

Ilustración 0-1 Diagrama GANTT

Bibliografía

[1] Java Sun
[http://java.sun.com/]

[2] Sitio Web Apache
[http://www.apache.org/]

[3] Sitio Web Xampp.
[http://www.apachefriends.org/es/xampp.html]

[4] Sitio Web Itext
[http://www.lowagie.com/iText/]

[5] Sitio Web Javamail
[http://java.sun.com/products/javamail/]

[6] Tutorial programación Servlets
[http http://www.programacion.com/tutorial/servlets_basico/]

[7] Tutorial avanzado Servlets
[http://www.programacion.com/java/tutorial/servlets_jsp/]

[8] Tutorial Itext
[http://itextdocs.lowagie.com/tutorial/]

[9] Tutorial Itext
[http://javaboutique.internet.com/tutorials/iText/]

[10] Tutorial Javamail.
[http://www.programacion.com/java/tutorial/javamail/]

[11] Tutorial Javamail con ejemplos
[http://www.chuidiang.com/java/herramientas/javamail/enviar-correo-javamail.php]

[12] Sitio web Netbeans.
[http://www.netbeans.org/]

[13] Sitio Web MySQL
[http://www.mysql.com/]

[14] Sitio Web Total Commander.
[http://www.ghisler.com/]

[15] Artículo comercio en Internet.
[http://ols.uas.mx/PubliWeb/Articulos/MRL-Internet_y_comercio.pdf]

[16] Artículo sobre EDI.
[http://www.csae.map.es/csi/silice/Svaedi4.html]

[17] Explicación sobre B2B.
[http://es.wikipedia.org/wiki/B2B]

[18] Artículo sobre comercio electrónico.
[http://es.wikipedia.org/wiki/Comercio_electr%C3%B3nico

[19] Introducción al comercio electrónico.
[http://www.monografias.com/trabajos12/monogrr/monogrr.shtml]

[20] Arquitectura orientada a servicios.
[http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios]

[21] Evolución del B2B.
[http://www.wharton.universia.net/index.cfm?fa=viewArticle&ID=364]

[22] B2B Marketing.
[http://www.monografias.com/trabajos38/business/business3.shtml#ecommer]

[23] Introducción EDI.
[http://www.monografias.com/trabajos/edi/edi.shtml]

[24] Explicación MVC.
[http://es.wikipedia.org/wiki/Modelo_Vista_Controlador]

[25] Utilización del MVC.
[http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista.shtml]

[26] Explicación sobre JSP..
[http://es.wikipedia.org/wiki/Java_Server_Pages]

[27] Tutorial JSP.
[http://geneura.ugr.es/~jmerelo/JSP/]

[28] Tutorial JSP y Servlets.
[http://www.programacion.com/java/tutorial/servlets_jsp/]

[29] Sitio Web Métrica V3
[http://www.csi.map.es/csi/metrica3/index.html]

[30] Aplicaciones Web en Wikipedia
[http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web]

[31] Pasado, Presente y Futuro de las aplicaciones Web
[http://www.slideshare.net/idumm/pasado-presente-y-futuro-de-las-aplicaciones-web

[32] Desarrollo en Cascada Wikipedia.
[http://es.wikipedia.org/wiki/Desarrollo_en_cascada /]

[33] Ciclo de vida Software.
[http://www.ia.uned.es/ia/asignaturas/adms/GuiaDidADMS/node10.html]

[34] Sitio Web Previgalia.
[http://www.previgalia.com/]

[35] Api Java
[http://java.sun.com/j2se/1.5.0/docs/api/]

[36] Api JavaMail
[http://java.sun.com/products/javamail/]

[37] Api Itext
[http://www.lowagie.com/iText/docs.html]

[38] Arquitectura Cliente - Servidor
[http://es.wikipedia.org/wiki/Cliente-servidor]

[39] Definición Arquitectura Cliente - Servidor.
[http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml]

[40] Tutorial Dreamweaver.
[http://www.programatium.com/dreamweaver.htm]

[41] Tutorial NetBeans.
[http://www.netbeans.org/kb/60/java/quickstart_es.html]

[42] Sitio Web de Prosinet.
[http://www.prosinet.com/]